AUTOMATIC DIGITAL COMPUTERS

by

M. V. WILKES
F.R.S.
Director of the University Mathematical Laboratory, Cambridge

NEW YORK
JOHN WILEY & SONS, INC.

First published in 1956

PRINTED IN GREAT BRITAIN BY
RICHARD CLAY AND COMPANY, LTD., BUNGAY, SUFFOLK

CONTENTS

1 The Development of Automatic Digital Computers . 1
2 The Principles of Logical Design . . 39
3 The Principles of Programme Construction 86
4 Relay Computers 133
5 Storage 152
6 Electronic Switching and Computing Circuits . . 210
7 The Design and Operation of Digital Computers . 257
Appendix: Machinery and Intelligence 289
Annotated Bibliography 294
Index . 301

PLATES

facing page

Fig. 1·1 Model constructed from parts made for Babbage's Difference Engine 4

Fig. 1·4 Part of the Mill of Babbage's Analytical Engine . 13

Fig. 1·7 General view of the Automatic Sequence Controlled Calculator 16

Fig. 1·17 General view of the ENIAC . 30

Fig. 1·23 General view of the EDVAC 37

Fig. 2·1 A photograph of the EDSAC taken shortly after its completion 39

Fig. 2·11 A parallel multiplier 64

Fig. 5·19 Photograph of a magnetic drum made by Ferranti Ltd. 181

Fig. 7·1 Early examples of packaged units for a small computer (Elliott Brothers (London) Ltd.) . . 268

PREFACE

THIS book is intended to provide a general introduction to the principles underlying the design and use of digital computers. It covers the subject now generally known as logical design without entering into a detailed discussion of electronic circuit techniques. It also deals with the way in which programmes are constructed, and methods by which the machine itself can be made to assist the programmer in his task. Discussion of what operations need to be programmed to solve particular problems—the subject of numerical analysis—is outside the scope of the book.

One or two machines are described in some detail, and references are made to a number of other machines in order to illustrate specific points. I have not, however, thought it necessary to give descriptions of all the machines mentioned. An annotated bibliography is provided for the benefit of readers who desire more information about particular machines or about any of the subjects discussed in the text. This bibliography is by no means complete, but many of the books and papers referred to in it themselves contain further lists of references. There are relatively few direct references in the text to the bibliography, and readers desiring information should consult the index where such references will be found.

A writer on a subject which has grown up rapidly finds himself in some difficulty in connexion with the choice of nomenclature. The object should be to exclude laboratory slang but to retain technical terms which are appropriate and widely understood. I was a little doubtful about the use of the word 'flip-flop' to describe a circuit with two stable states. This word is, however, in wide use and is less cumbersome than the alternatives of 'trigger circuit' or 'bi-stable circuit'. Some readers may be accustomed to the use of the term 'flip-flop' to indicate a mono-stable circuit, but I can assure them that I have used it in the sense in which it is generally used in computer circles.

Notation for block diagrams also presents a difficulty. Most groups which have worked actively on the design of computers have developed their own notation. All such notations bear a family resemblance but show evidence of having been influenced by the logical structure of the machines they have been evolved to describe. A number of official and semi-official bodies have produced recommended lists of symbols, but it cannot be said that anything like standardization has yet been achieved. After some thought I decided that I could simplify my own task, without putting difficulties in the way of my readers, by adopting the system of notation in current use in my own laboratory. I was helped in reaching this decision by the fact that most of the block diagrams in this book are relatively simple and contain only a few logical elements.

Logical design is a no-man's land in which practical engineers and those whose training has been purely theoretical can meet on approximately equal terms. I hope, however, that nothing I have written will lead any members of the latter class to think that they can design a digital computer by themselves, or to press their opinions too strongly on matters in which practical engineering experience and judgement are necessary. In the end the last word should rest with the engineer who has overall responsibility for making the machine work.

Anyone writing on digital computers cannot help being very conscious of how much the subject owes to the free interchange of information and ideas which has taken place between the various design groups. This book is the outcome of such interchange, and I would like to express my indebtedness to my many friends and colleagues on other projects, particularly to those in the United States. In addition, I owe a special debt to the colleagues with whom I have worked closely in Cambridge during the past nine years.

A large part of the work of preparing the diagrams and of editing the manuscript for the Press has been undertaken by Mr E. N. Mutch. I am very grateful to him for taking this heavy load off my shoulders, and for the constant encouragement and sound advice he has given me.

I am also very grateful to Miss M. O. Lewin, who read the book, first in manuscript and later in proof, and pointed out a number of errors. I am much indebted to Mr J. Leech for a very careful reading of the proofs, and to Dr J. C. P. Miller for advice on a number of points.

I would like to express my thanks to the Royal Society of Arts for permission to make use, in the early chapters, of material presented to the Society in 1951 in three Cantor lectures, to the Controller of H.M.S.O. for permission to reproduce the photograph in Fig. 1·4 (Crown copyright: from an exhibit in the Science Museum, South Kensington), to the American Institute of Electrical Engineers for permission to reproduce Figs. 1·15 and 1·16 from a paper by J. G. Brainerd and T. K. Sharpless published in *Electrical Engineering*, vol. 67, p. 163 (1948), to the Commanding General, Ballistic Research Laboratories, Aberdeen Proving Ground, Maryland, for providing photographs of the ENIAC and the EDVAC and for allowing me to reproduce them in Figs. 1·17 and 1·23, to Ferranti Ltd., for providing the photograph shown in Fig. 5·19, and to Elliott Bros. (London) Ltd., for providing the photograph shown in Fig. 7.1.

M. V. WILKES

The University Mathematical Laboratory,
Cambridge.
September, 1955.

CHAPTER 1

THE DEVELOPMENT OF AUTOMATIC DIGITAL COMPUTERS

AUTOMATIC digital computers resemble the familiar desk calculating machines in that they handle numbers expressed as a series of digits, in much the same way as one handles them when doing arithmetic with paper and pencil. Automatic machines differ, however, from desk machines in that they can perform long sequences of arithmetic operations without the intervention of an operator and not just one addition, subtraction, multiplication, or division at a setting. This automatic feature enables the great speed of which electromechanical and, more especially, electronic devices are capable to be fully utilized.

In addition to the digital computers which will be discussed in this book, various large-scale calculating machines working on the *analogue* principle have been built. In these machines numbers are expressed by the measure of some physical quantity and the result is obtained by making a measurement. Of these machines the best known is perhaps the differential analyser, which is an analogue machine for solving ordinary differential equations. In the original differential analyser, described by V. Bush in 1931, numbers were represented by the angular rotation of shafts, but the name has since been applied to other analogue devices in which electrical voltages or currents are used instead. Discussion of these machines, however, is outside the scope of this book.

It is very important to realize that a digital calculating machine can only be used to solve problems which are expressed in an arithmetic form, that is, problems which are stated precisely as a series of arithmetic operations—additions, subtractions, multiplications, and divisions—and in which all numbers required for starting are specified. Some problems, such as those which arise in statistics or in commercial accountancy, are in this form to begin with. Many mathematical problems, however, require transformation before the machine can be used to solve them. For example, an integral must be expressed approximately as the sum of a finite number of quantities before it can be evaluated on a machine. People who have not fully appreciated the way in which a digital machine works sometimes ask questions such as the following: 'Can a digital calculating machine enumerate the prime numbers between specified limits?' Questions of this kind must always be answered by another question: 'Can the results required be obtained by a sequence of arithmetic operations?'

If the answer to this question is 'Yes', then the problem can in principle be solved on a digital computing machine.

Chapters 2 and 3 of this book contain a discussion of the detailed steps which must be taken to make a machine carry out the arithmetic operations necessary to solve a problem, when once it has been decided what those operations are. The wider problem of turning a mathematical problem into an arithmetic one is dealt with in books on numerical analysis. If the problem were to be solved by a human computer—assisted perhaps by a desk calculating machine—it would be possible to draw up for him a series of *orders*, or *instructions*, each calling for a simple arithmetic or copying operation. These orders he would execute one after the other, and he would not need to have any understanding of the mathematical significance of the steps he was taking. A problem is communicated to an automatic calculating machine in a similar way, except that the orders, or instructions, are presented in some form which can be sensed automatically—for example, by means of a punched paper tape. The sequence of orders is known as the *programme*, and the machine performs it automatically without intervention by the operator. An automatic calculating machine thus provides a mechanization of a human computer plus a desk calculating machine—although, of course, it is only the computer's ability to obey rules which is mechanized, not his intelligence. More is said about the subject of intelligence in the Appendix.

BABBAGE'S CALCULATING MACHINES

The first truly automatic calculating machine was not finished until 1944, so that the subject is a modern one. Part of this chapter will, however, be devoted to an account of the work of Charles Babbage, who had ideas for such a machine over a hundred years ago. Babbage's ideas are worth studying for their historical interest, and they will also be found to form an ideal introduction to the more modern developments of the subject.

Charles Babbage was born on 26th December, 1792, at Totnes in Devonshire, and he went up to Trinity College, Cambridge, in 1810. This was a time of stagnation for mathematics at Cambridge, which had become cut off from the developments in mathematical analysis taking place on the Continent. Babbage found himself out of sympathy with the current teaching and did not take the Mathematical Tripos; this, however, did not prevent him from becoming Lucasian Professor in 1828 and holding the chair for eleven years, although without giving any lectures. As an undergraduate, Babbage was, with Peacock and Herschel, a founder member of the Analytical Society, which was formed, he said, to inculcate 'the Principles of pure D-ism in opposition to the Dot-age of the university', this being a reference to the continued use of the fluxion notation of Newton in-

stead of the more flexible notation of Leibnitz (e.g., $\dot{x}$ instead of dx/dt). It was in the rooms of the Analytical Society in 1812 or 1813 that Babbage had his first idea for a computing machine.

The Difference Engine

During the time of the First Republic an elaborate project had been started in France for the computation of mathematical tables designed to facilitate the general use of the metric system.* This project was organized on what we should now call production lines, and the staff were divided into three grades. First, there were some five or six mathematicians who decided on the best mathematical methods and formulae to be used. Secondly, there were eight or ten computers who were competent to handle these formulae and to compute numerical values from them; their role was to compute 'pivotal values', that is, selected values of the function spaced at five or ten times the interval required in the final table. Thirdly, there were computers of a lower grade, nearly 100 in number, who understood only the elements of arithmetic, but who were able, by following rules laid down, to perform the final stage of the tabulation. This was to fill in the gaps between the pivotal values—a process of systematic interpolation known as subtabulation. It was this last group of computers that Babbage proposed to replace by his Difference Engine, which was a machine designed to tabulate polynomials by the method of differences. This method is well known in numerical analysis, and it is only necessary to give a brief description of it.

A difference engine needs to have a number of registers—one more than the degree of the polynomial—in which numbers can be stored, and to be able to add successively the number in each register to that in an adjacent register. For example, a difference engine designed to tabulate polynomials of the second degree would need three registers. Suppose for simplicity the polynomial to be tabulated is $f(x)=x^2$ and that at the start the registers contain the numbers:

1 1 2

During the first stage of the operating cycle of the machine the number in the second register is increased by the addition of the number in the third register. At the end of this stage the numbers are:

1 3 2

During the second stage the number in the second register is added to that in the first, so that the numbers become:

4 3 2

* Seventeen folio volumes of tables were computed for this project but were unfortunately never published. In 1820 the British Government made a proposal that they should be published jointly but nothing came of it.

If these two stages are repeated, the numbers in the registers at the end will be:

9 5 2

It will be seen that the numbers successively obtained in the first register constitute a sequence of squares. By taking suitable starting values for the numbers in the three registers, any polynomial of the second degree can be tabulated by this process. A difference engine thus needs storage registers for numbers and a mechanism whereby additions may be performed according to a fixed cycle.

Babbage made a small machine which would tabulate to eight decimals a function whose second differences were constant, for example, x^2+x+41. He took steps to obtain official support, and in 1823 a machine with provision for 20 decimals, and capable of tabulating a function whose seventh differences were constant, was begun for the British Government. This machine was intended to print its results, for Babbage put great stress on this as a means of avoiding errors of transcription. The machine was never completed, work being stopped in 1833 and the project abandoned in 1842, although a part of the machine was finished and exhibited in 1862. This part is now in the Science Museum at South Kensington, London.

In all, the Government spent about £17,000 on the Difference Engine. It is a pity that so great a project was not brought to completion. The immediate cause of the stopping of work was that Babbage had financial differences with his engineer, who resigned from the project, retaining—as he apparently had a right to do—the specially constructed tools, and dismissing the men he had been employing to work on it. This was a serious matter, since engineering technique was not at that time equal to the demands that the construction of Babbage's machine was making on it, and new techniques had to be developed. It must be remembered that in Babbage's time even the simpler forms of desk calculating machine were not in general use, although Pascal had demonstrated an adding machine in 1642, and Leibnitz a machine which could multiply as well as add in 1694. The first really practical machine was the arithmometer designed by Thomas de Colmar in 1820, and this enjoyed a limited commercial success, although it was not a very convenient machine to use. The real commercial development of desk calculating machines did not, however, begin until the end of the century, when at about the same time Odhner produced the first of the barrel-type machines (of which the Brunsviga is the best known today) and the Millionaire was put on the market. In producing a machine capable of carrying out a programme, and not simply additions or multiplications one at a time, Babbage was, therefore, attempting to take two steps at once. One incidental result of Babbage's work was to help the advancement of

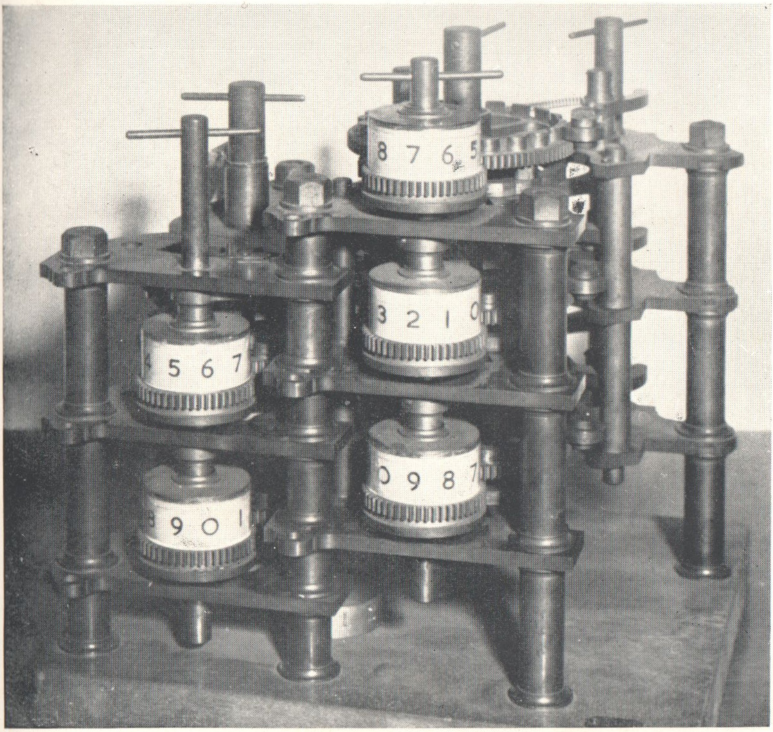


FIG. 1·1.—MODEL CONSTRUCTED FROM PARTS MADE FOR BABBAGE'S DIFFERENCE ENGINE

mechanical engineering, and in this connexion it is interesting to record that Whitworth received his early training on Babbage's Difference Engine. After Babbage's death, his son constructed a number of demonstration models from parts of the Difference Engine. One of these was sent to Cambridge, and a photograph of it is shown in Fig. 1·1.

In 1834 Dr Lardner published in the *Edinburgh Review* an article describing in some detail the principles on which Babbage's Difference Engine worked. This had the effect of stimulating Georg Scheutz, a Swede, to experiment with the design of a machine for the same purpose. With the aid of his son, Edvard Scheutz, he constructed a working model which was demonstrated to the Royal Swedish Academy of Sciences in 1843. After some difficulty he succeeded in obtaining a grant from the Swedish Government for the construction of a machine working with four differences and having capacity for 14 decimals. This machine was very successful, and it was demonstrated in London and Paris in 1855.

Babbage encouraged Georg and Edvard Scheutz and gave publicity to their machine. The original was purchased for $5,000 by an American business man and presented to the Dudley Observatory at Albany, N.Y. Unfortunately, as far as is known, this machine was never put to any practical use. However, a copy was made to the order of the British Government and used in the Registrar-General's office for the preparation of a set of life-tables published in 1864 under the name of William Farr, then Registrar-General. Some of the tables in this book were printed from blocks cast from stereo moulds prepared directly by the machine. Apart from being subtabulated by machinery, these tables were, as far as I know, the first in which the setting of type by hand was eliminated, together with the opportunities for error which this allows.

The British copy of Scheutz's machine is now in the Science Museum at South Kensington. It is quite different in mechanical design from Babbage's machine, and its much lighter construction gives it altogether a more modern appearance.

Subtabulation is nowadays performed mechanically either by a National accounting machine or by punched-card equipment. These machines produce a printed table rather than stereo moulds. The printed table can be reproduced photographically, although, until recently, few tables have been treated in this way. Modern techniques seem to have been applied more assiduously in reducing the labour of computation rather than that of type-setting and proof-reading. The coming into general use of photo offset methods of printing is, however, undoubtedly leading to a rectification of this situation. A good example has been set by the Computation Laboratory of Harvard University, who have made much use of photographic methods in the

printing of mathematical tables. Until, however, the need for spotting and retouching the negatives is eliminated, the use of photo offset methods will not entirely remove the danger of errors being introduced during printing.

National and punched-card machines are also used for differencing, a process which it is now appreciated is as useful to a computer as subtabulation.

The Analytical Engine

A factor which contributed to the decision not to restart work on the Difference Engine was that Babbage had a new idea, this time for a much more elaborate machine, which he called an Analytical Engine. Although the two machines were quite independent, this had the effect of confusing the issue at a critical moment, particularly as Babbage conceived it to be his duty to point out that the new mechanism he was designing for the Analytical Engine would enable a much simpler difference engine to be made.

The Analytical Engine was never built, but it is most interesting from the present point of view, since it was to be a completely automatic computing machine with all the facilities which we recognize in these machines today. A description of it will form a foundation for a discussion of modern automatic computing machines.

An automatic computing machine must have the following components: (1) a *store* for numbers—Babbage used that word, although the part of the machine concerned is now often called the *memory*; (2) a device for performing arithmetic operations on those numbers—Babbage called this a *mill*—the modern name is *arithmetic unit*; (3) a device for causing the operations of the machine, including the transfer of numbers from one place to another, to take place in their correct sequence—Babbage had no name for this, but it is now usually referred to as the *control unit*; (4) input and output devices.

For the store, Babbage proposed to use columns of wheels, each wheel being capable of resting in any one of ten positions and, therefore, of storing a single decimal digit. When numbers were to be transferred from the store to another part of the machine, a set of racks would be engaged with teeth cut in the wheels, and each rack would be driven until the wheel concerned reached its zero position. The motion would be transmitted by rods and links to the arithmetic unit, where, by means of another rack, it would be used to move one of the wheels of a register to the correct position. Babbage gave a lot of thought to the design of the arithmetic unit, particularly to the problem of how to effect the carry-over, or tens-transmission, as rapidly as possible. Something is said about his ideas on this subject below.

For the control, Babbage proposed to adapt the mechanism used

in a Jacquard loom for the production of fabrics having complicated patterns. The principle on which this mechanism operates is illustrated in Fig. 1·2. Before each passage of the shuttle a group of warp threads, selected according to the requirements of the pattern, must be lifted so that the shuttle may pass underneath. The various threads pass through wire eyes (healds) which are connected by strings to a set of hooks arranged in a framework, several threads being attached to each hook; any particular group of warp threads, therefore, can be

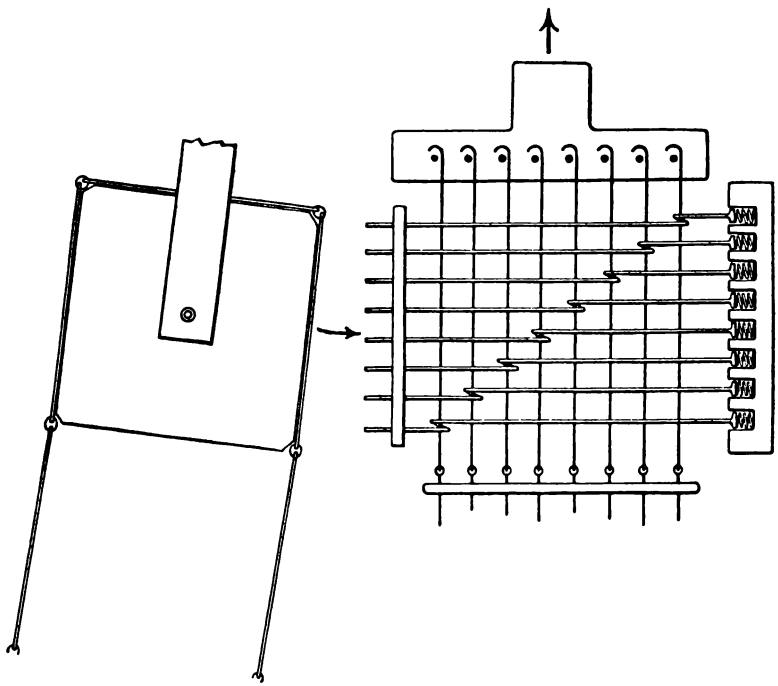


Fig. 1·2.—Principle of operation of the Jacquard mechanism.

lifted by lifting the corresponding hook. The lifting of the hooks is accomplished by means of a set of horizontal bars which rise just before the shuttle passes.

The hooks which are to be lifted during any particular passage of the shuttle are specified by holes punched in a set of cards, one card to each passage. The cards are strung together with loops of string so as to form a continuous chain, and pass over a cylinder having a square cross-section. Just before the shuttle passes, the cylinder moves bodily to one side so that one of the cards is pressed against a matrix of horizontal rods, there being one rod associated with each hook.

The faces of the cylinder have clearance holes drilled in them so that only those rods which come into contact with the card are moved, all those rods which are opposite holes punched in the card remaining at rest. The rods which move engage with the corresponding hooks and move them out of range of the rising bar. After the passage of the shuttle, the cylinder returns to its normal position and rotates so that a new card is presented to the matrix of rods. In an ordinary Jacquard loom the matrix may contain 400 or more rods and the corresponding number of hooks arranged in several rows.

The use of a moving cylinder with a chain of cards was introduced by Falcon in 1728, and in 1745 Vaucanson contributed the idea of a moving bar which lifts the hooks. Jacquard appears to have combined the previous inventions and to have made the machine a commercial success. A design produced by him in 1804 was soon in widespread use in France and came into general use in England in about 1824.

In Babbage's drawing-room there was hanging a woven portrait of the inventor Jacquard made on one of his own looms at Lyons in 1839, a few years after his death.* Babbage tells an interesting story about this portrait in connexion with his first meeting with the Prince Consort. Like many other Englishmen of his time, Babbage did not feel very kindly disposed towards the Prince, and he had felt rather slighted by certain events which had occurred during the preparations for the Great Exhibition of 1851. Accordingly, he was not very pleased when one day he received a message from the Duke of Wellington to the effect that the Prince wished to meet him and to hear about the Analytical Engine. Babbage could not refuse to receive the Prince, but he resolved to treat him with the deference his rank demanded but no more. When the Duke and the Prince entered Babbage's drawing-room and were shown the portrait of Jacquard, the Duke took it for an engraving; the Prince, however, had seen a copy before, knew what it was, and appreciated its significance. This at once caused Babbage's heart to warm towards him and, putting aside his former prejudice, he decided to be as forthcoming as possible in his explanations of the Analytical Engine. This story, in so far as it concerns the Prince Consort, could no doubt be paralleled many times.

Babbage's idea appears to have been to have two Jacquard mechanisms, each containing a string of cards, to control the action of his machine. One mechanism —which would contain 'operation' cards— would be connected to the arithmetic unit and would set this unit in a state of readiness to perform a multiplication, addition, subtraction, or division, according to the holes punched on the card being sensed.

* Examples of this portrait are in various museums, including the Science Museum, London, and the Smithsonian Institution, Washington, D.C.

The other mechanism would control the transfer of numbers to and from the store. Fig. 1·3, reproduced from Babbage's book, shows how he proposed to punch his operation cards.

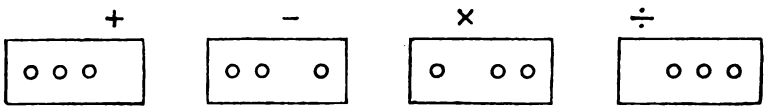


Fig. 1·3.—The method of representing operations on punched cards proposed by Babbage (from *Passages from the Life of a Philosopher*).

Provision was also to be made for supplying numerical information by means of punched cards when required. In this way logarithmic or other tables could be made available to the machine if necessary. The idea was that when a tabular entry was required the machine would ring a bell and display the argument in a window. The attendant would then fetch the card and place it in the machine. The machine would verify that the argument punched on the card was the one required and would proceed with the calculation. (Babbage says that if the attendant put in a wrong card the machine would ring a louder bell.)

In connexion with the use of tables Babbage makes the following remark: 'It will be an interesting question, which time only can solve, to know whether such tables on cards will ever be required by the Engine. Tables are used for saving the time of continually computing individual numbers. But the computations to be made by the Engine are so rapid that it seems most probable that it will make shorter work by computing directly from proper formulae than by having recourse even to its own Tables.' This is certainly true for modern high-speed automatic calculating machines and would possibly have been true for the Analytical Engine also, although this is doubtful in view of its slowness compared with modern machines. That Babbage should have considered the question and realized that only experience could provide the answer illustrates how sound his judgement was on matters of principle; he could take what Hartree has called 'the machine's eye view' of numerical calculations.

Three methods of output were envisaged: the machine could be used either (i) to print one or two copies of a table, (ii) to make a stereotype mould, or (iii) to punch cards. The last method of output would be used to enable the machine to calculate its own tables (if these were required) and to punch them on cards. For input, Babbage seems to have contemplated that the numbers required for starting would be set by hand on the wheels of the store, or on one or more registers of the mill; they could also be supplied by means of the punched-card mechanism used for tables.

The times taken for various operations were estimated as follows: addition or subtraction, 1 second; multiplication (50 decimals by 50 decimals), 1 minute; division (100 decimals by 50 decimals), 1 minute. The last two times would be reduced if the numbers contained fewer digits. Babbage proposed that the store should have capacity for 1,000 numbers, each number to have 50 decimals, 'in order to be somewhat in advance of the greatest number that may ever be required'. Later in his life, according to a British Association report, Babbage contemplated 200 numbers of 25 decimals. His son, H. P. Babbage, in a paper presented to the British Association in 1888 (after Babbage's death), speaks of 20 numbers of 25 decimals for a first machine. It will be interesting to compare these capacities with what modern designers of automatic computing machines have provided. There is, today, the same conflict between what is theoretically desirable and what is mechanically, or economically, practicable.

Babbage proposed that the algebraic sign of a number would be indicated by a separate wheel not connected by tens-transmission with the others. This wheel was to be identical with the others in the register, and if it showed an even number, or zero, the sign was to be taken as positive; otherwise it was to be taken as negative.

Programming for the Analytical Engine

Now that a description has been given of the main features of the Analytical Engine, it remains to show how it could be used for mathematical calculation. Babbage himself wrote only in general terms on this subject, but fortunately some of his views have come to us indirectly. In 1840 Babbage was invited to visit Italy to discuss his ideas with Italian mathematicians. Among those present was a young officer of the military engineers, L. F. Menabrea, who was at that time on the staff of the Military Academy in Turin. Menabrea wrote an account of what he learned from Babbage and published it in French in a Geneva journal in October 1842. Menabrea afterwards rose to the rank of General and had a distinguished career in politics and diplomacy; he continued to contribute from time to time to scientific periodicals.

Menabrea's paper was translated into English by the Countess of Lovelace, who was apparently well acquainted with Babbage. At his encouragement, she wrote some notes of her own, and the translation and notes (the notes being almost three times as long as the original paper) were published in Taylor's *Scientific Memoirs* in 1843. This paper and the notes are concerned only briefly with the mechanical aspects of the Engine, but deal with the subject now called programming.

Lady Lovelace was the only child of Lord and Lady Byron. She was evidently a woman of some ability and she had a clear grasp of

her subject. If the Analytical Engine had been built she would have been well qualified for the post of leading mathematician.

One example of programming given by Menabrea is the solution of the simultaneous equations

$$mx + ny = d,$$
$$m'x + n'y = d',$$

where, of course, it is assumed that numerical values are given for m, n, d, m', n', and d'. The problem is reduced to a piece of simple arithmetic by writing the equations in the form

$$x = \frac{dn' - d'n}{n'm - nm'},$$
$$y = \frac{d'm - dm'}{n'm - nm'}.$$

Menabrea denotes the contents of the various storage registers by V_0, V_1, &c., and assumes that they are initially as follows:

$V_0 = m$ $\qquad$ $V_3 = m'$
$V_1 = n$ $\qquad$ $V_4 = n'$
$V_2 = d$ $\qquad$ $V_5 = d'$

The various steps of the programme necessary for the calculation of x are then as given in Table 1·1. The steps necessary for the calculation of y follow similar lines.

TABLE 1·1

Number of the operations	Operation-cards. Symbols indicating the nature of the operations	Cards of the variables		Progress of the operations
		Columns on which operations are to be performed	Columns which receive results of operations	
1	×	$V_2 \times V_4 =$	V_8	$= dn'$
2	×	$V_5 \times V_1 =$	V_9	$= d'n$
3	×	$V_4 \times V_0 =$	V_{10}	$= n'm$
4	×	$V_1 \times V_3 =$	V_{11}	$= nm'$
5	−	$V_8 - V_9 =$	V_{12}	$= dn' - d'n$
6	−	$V_{10} - V_{11} =$	V_{13}	$= n'm - nm'$
7	÷	$V_{12}/V_{13} =$	V_{14}	$= x = \frac{dn' - d'n}{n'm - nm'}$

One of the most striking features of Babbage's Analytical Engine is the way in which what we should now call conditional operations were to be handled. Babbage introduces this subject by saying that one of the Italian mathematicians, Professor Mosotti, came to him one day with the following difficulty: 'He remarked that he was now

quite ready to admit the power of mechanism over numerical, and even over algebraical relations, to any extent. But he added that he had no conception how the machine could perform the act of judgement sometimes required during an analytical inquiry,* when two or more different courses presented themselves, especially as the proper course to be adopted could not be known in many cases until all the previous portion had been gone through. I then inquired whether the solution of a numerical equation of any degree by the usual, but very tedious proceeding of approximation would be a type of the difficulty to be explained. He at once admitted that it would be a very eminent one.'

Babbage had in mind a method similar to the one now known as Horner's method. He showed that the decision as to which of two courses of action is to be taken depends on whether some calculated quantity comes out to be positive or negative. If it comes out negative it will be because a large number has been subtracted from a small one, and he considers, by way of example, what happens in a machine with a capacity for 20 decimals if 511 is subtracted from 423, thus:

00000	00000	00000	00423
00000	00000	00000	00511
99999	99999	99999	99912

The action of the tens-transmission causes 9s to appear successively in the decimal places to the left of those containing significant digits. The 9s are, as it were, propagated to the extreme left of the register and would go further if the register were extended. Babbage points out that the motion of the carrying lever which would cause a 9 to appear in the 21st place, if this place existed, may be used to set any desired action in train. Since this lever only moves if the result is negative, whether the action takes place or not is conditional on the sign of the number calculated. In his choice of the action to be initiated by a conditional operation Babbage again showed his characteristic soundness of judgement. He proposed that it should be used to advance or roll back the cards on the Jacquard mechanisms to any specified extent (specified presumably by a number in a register or punched on one of the variable cards). If the cards are advanced, a section of the programme will be skipped; if they are rolled back, a section of it will be repeated. In this way, a cycle of operations can be repeated until the answer at the end changes sign, when another cycle of operations can be begun.

In the example given by Babbage the number whose sign changes is one which occurs naturally in the calculation. Lady Lovelace quotes a more complicated example in which the number which changes sign

* That is, numerical calculation.

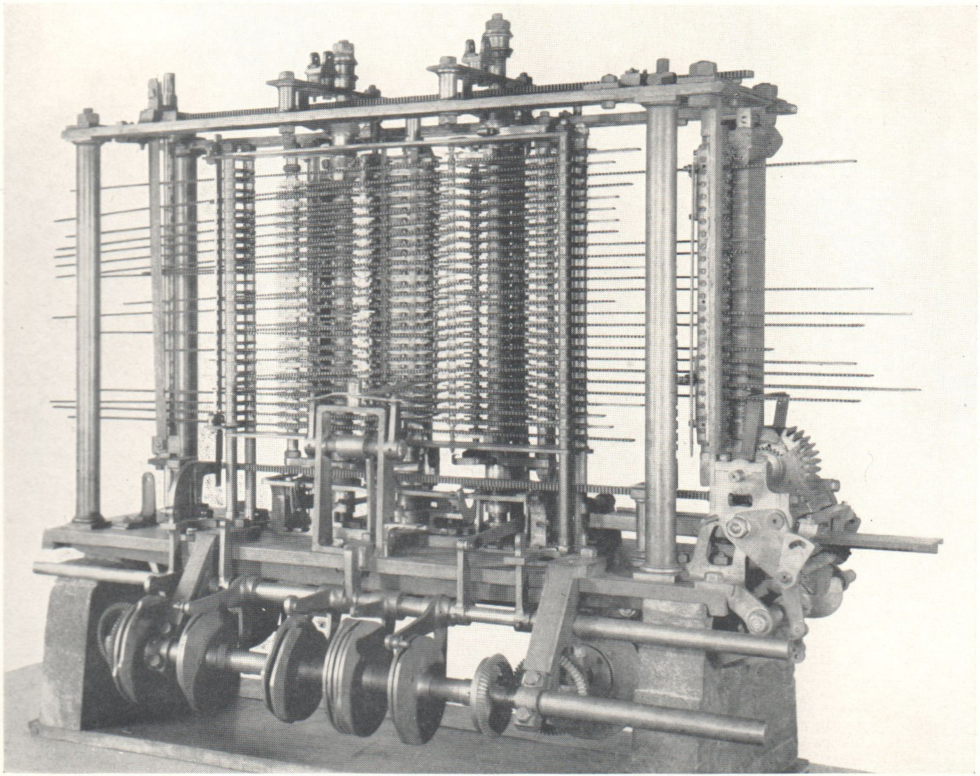


FIG. 1·4.—PART OF THE MILL OF BABBAGE'S ANALYTICAL ENGINE

is one introduced specially for that purpose. The example is the calculation of Bernoulli numbers from a recurrence formula—a very ambitious piece of programming indeed. In the calculation of the nth Bernoulli number B_n a certain cycle of operations has to be performed $n+1$ times. It is arranged that at the beginning of the calculation of B_n a certain storage register contains the number n and operations are included in the programme which cause unity to be subtracted from the number in this register each time the cycle is performed. Thus, at the end of successive repetitions of the cycle the register contains the numbers $n-1$, $n-2$, &c. After n repetitions the number is zero, and after the next repetition it becomes negative. A conditional operation, therefore, inserted at the end of the cycle, will cause the machine to go ahead with the next part of the calculation after $n+1$ repetitions of the cycle, instead of repeating the cycle again. The allocation of a storage register to hold a number which is used solely for counting is a prominent feature of the technique used in preparing programmes for modern automatic calculating machines.

The question of conditional operations will not be pursued further here, since it will come to the fore in a later chapter when programming for modern machines is discussed. The importance of the principle will, however, be realized. It enables calculations to be programmed even if the number of steps cannot be determined until the calculation has proceeded some way, or if alternative procedures have to be used at certain points of the calculation according to the values assumed by the variables.

Mechanical Design

Babbage himself did not publish many details of the mechanical design of the Analytical Engine, although some notes were included in a lecture given to the British Association by H. P. Babbage in 1888. Babbage left detailed drawings of the Analytical Engine, and these, together with his notebooks and a portion of the mill put together in 1906 by his son, are now in the Science Museum at South Kensington. Fig. 1·4 shows part of the mill, and comparison with Fig. 1·1 will show that its construction is considerably lighter than that of the Difference Engine.

One of Babbage's mechanical achievements is especially worth recording, since the principle can be applied to many different types of machine. It is what he called *anticipatory carry*. In the mechanism designed for the Difference Engine adding takes place in two phases which might be called the *add phase* and the *carry phase* respectively. Suppose that the numbers 12,543 and 21,608 are to be added together and that initially the numbers are stored on two sets of five counter wheels arranged alongside each other. During the add phase corresponding wheels in the two sets are connected together by means of

meshing teeth. The wheels of the first counter are then rotated backwards until they reach zero. At the end of this phase the second counter will contain the number 33,141, and in order to turn this into the correct sum carry digits must be added into the second and fourth places. This is done by arranging that whenever a wheel in the second counter passes from 9 to 0 during the course of the add phase a catch is released; during the subsequent carry phase the various catches are reset by a separate motion and, in the process of resetting, the wheel corresponding to the next higher place is rotated one step. Since this may make the last-mentioned wheel move from 9 to 0 and release the corresponding catch, it is clear that resetting of the catches must take place in succession, starting at the least significant end of the register. The system is therefore known as *successive carry*. If there are 50 places it is found that the time required for the carry phase may be up to five times that required for the add phase. The Brunsviga calculating machine uses the system of successive carry, and with 15 decimal places the same length of time is allowed for the carry and the add phases. Babbage's system of anticipatory carry was designed to speed up the process of addition by reducing the time required for carry, and hence to increase the overall operating speed of the machine.

It is perhaps easier to describe the principle on which anticipatory carry works if we imagine that the counter wheels are electrically operated and can be stepped on from one position to the next by applying a pulse to an input terminal. Fig. 1·5 then indicates the

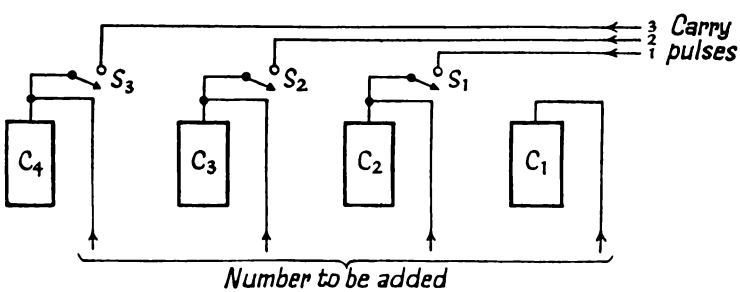


Fig. 1·5.—Successive carry. C_1, C_2, C_3, C_4 represent counter wheels and S_1, S_2, S_3, S_4 represent the associated switches. Each switch moves up and stays up if the corresponding counter wheel passes 9.

system of successive carry. A switch, normally in the position shown, is associated with each counter wheel; this switch moves upwards (and stays there) if the wheel passes from 9 to 0. During the add phase, pulses representing the 'incident' number are applied to the input terminals of the various counter wheels; during the carry phase single pulses are applied in succession to the leads marked 1, 2, 3, &c.,

sufficient time being allowed between them for a switch to move if the corresponding counter wheel passes from 9 to 0.

In the system of anticipatory carry which Babbage invented, carry into all stages takes place at the same time. The principle is shown in Fig. 1·6. A two-position switch, normally in the position shown, is associated with each counter wheel; it moves upwards (and stays

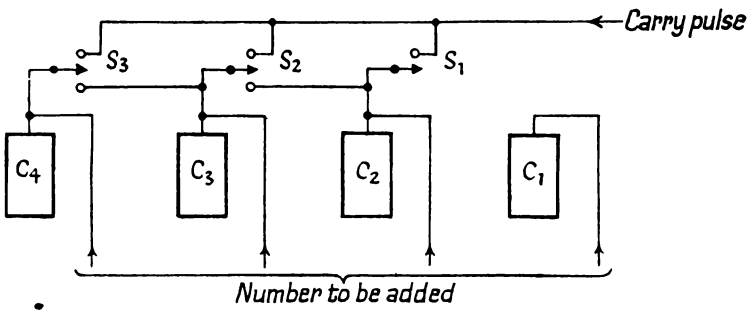


Fig. 1·6.—Anticipatory carry. Each switch moves up and stays up if the corresponding counter wheel passes 9; it moves down if the counter wheel stops on 9.

there) if the associated counter passes from 9 to 0 and moves downwards if the counter stops on a 9. It will be seen that these two conditions are mutually exclusive. A single pulse applied to the carry line then causes the correct carry to take place into all stages. The time taken for carry is thus independent of the number of stages.

Babbage attached great importance to his invention of anticipatory carry, and there is no doubt that it was a marked advance. The principle lends itself more readily to use in electrical or electro-mechanical adders than in mechanical ones, and it is used, for example, in IBM and Hollerith punched-card equipment.

LATER HISTORY

Babbage died in 1871. The Analytical Engine had been designed in considerable detail on paper, and Babbage had spent a considerable sum on it, but only a small part had been constructed. His son continued to keep the project before the public, and did a certain amount of work on the adding and subtracting mechanism of the mill. This he finally made to work, and it is now to be seen in the Science Museum at South Kensington. No support was, however, forthcoming for the completion of the machine.

At least one other proposal for the construction of an automatic calculating machine was made—by P. E. Ludgate in 1909—but as far as I am aware nothing was ever built. At about the same time a kind

of calculating typewriter was demonstrated by Torres y Quevedo of Madrid. This, however, would only perform single additions, subtractions, multiplications, or divisions, and there was no question of developing it so as to be capable of performing a programme. The designer was interested in the machine, not so much because of any use it might have had as an aid to practical computation, but because it provided an illustration of the fact that an automatic device can be designed to perform operations which involve some kind of discrimination. The machine consisted of a typewriter connected electrically to an ordinary desk calculating machine for which the motive power was provided by an electric motor working through magnetically operated clutches. The calculation to be performed was set on the machine by the operator typing, for example,

$$352 \times 257 =$$

after which the machine would perform the calculation and fill in the answer.

THE AUTOMATIC SEQUENCE CONTROLLED CALCULATOR

After Babbage's time desk calculating machines of both hand and electrically operated types became developed to a high degree of reliability, as did also machines using punched cards for accounting and statistical purposes. Babbage's work became forgotten, and in 1937 it occurred independently to Howard H. Aiken, now Director of the Computation Laboratory of Harvard University, that the components and techniques developed in connexion with punched-card machines would enable a completely automatic calculating machine to be built. He therefore approached the International Business Machines Corporation, who took the idea up with enthusiasm, and as a result the Automatic Sequence Controlled Calculator was formally presented to Harvard University by the Corporation on 7th August, 1944. A general view of the machine is given in Fig. 1·7.

The Automatic Sequence Controlled Calculator—or Mark I as it is often called at Harvard—is not a punched-card machine, although punched-card input and output mechanisms are provided. It was built as far as possible with the aid of standard components from IBM equipment. The principle of operation is mechanical, although relays and clutches operated by electro-magnets are used.

As in Babbage's Analytical Engine, numbers are stored in registers composed of wheels, each wheel capable of taking up ten distinct positions; a difference, however, is that each register in the store is fitted with tens-transmission and is, therefore, a complete adding machine. It is convenient to refer to such a register as an accumulator register, or simply as an accumulator. A number standing in one accumulator can be passed to another accumulator and added to the

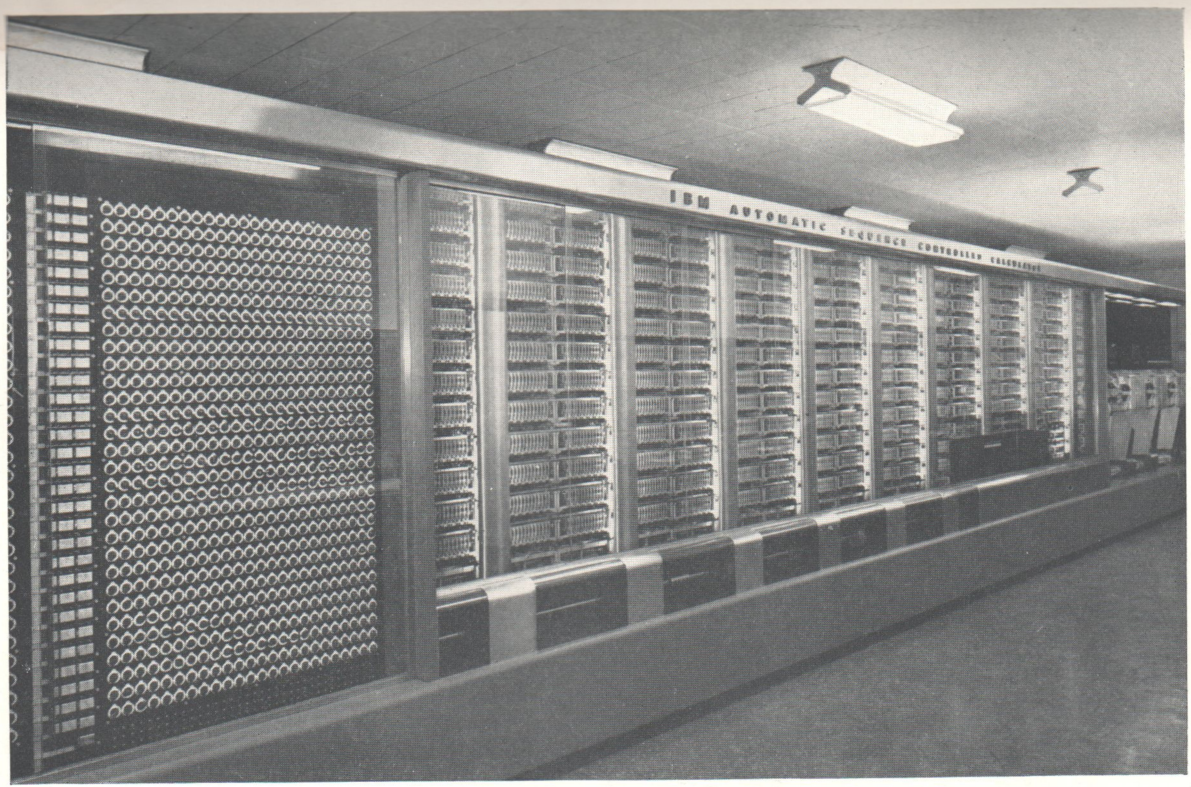


FIG. 1·7.—GENERAL VIEW OF THE AUTOMATIC SEQUENCE CONTROLLED CALCULATOR

number therein without passing through an arithmetic unit. A separate unit for multiplication and division is, however, provided. The accumulators are all fitted with an anticipatory carry system similar in principle to that devised by Babbage.

There are 72 accumulators in the store. Each is capable of holding a number of 23 decimals. In addition, there are 60 sets of switches on which constants may be set by hand. These constants may be used by the machine during the course of a calculation, but their values cannot be changed by the machine itself.

The operations of the machine are carried out under the control of a paper tape on which the orders, or lines of coding as they are usually called at Harvard, are punched. There are 24 positions across the tape in which holes can be punched, and the rows are sensed one at a time by a set of 24 peckers, that is, small rods which feel for holes in the tape and cause electrical contacts to be made when holes are present. When each operation has been performed the tape is stepped on so that the next row of holes comes under the peckers. It will be seen that punched tape plays exactly the same role in the Automatic Sequence Controlled Calculator as the sets of Jacquard cards strung together were to play in Babbage's Analytical Engine. In addition to the tape used for the programme there are other tapes on which numerical quantities, such as values of functions, may be punched.

The principle of operation of the machine may be briefly described as follows. Each group of counter wheels forming an accumulator has running through it a shaft which rotates continuously at 200 r.p.m. Wheels can be connected to this shaft for any desired portion of a revolution by means of magnetically operated clutches. A brush running over 10 segmental contacts is fitted to each wheel so that an electrical indication of the number standing in an accumulator is available. When a number standing in one accumulator is to be added to that standing in another, connexions are set up between the brushes of the first counter and the clutch mechanisms of the second in such a way that the wheels of the second counter are connected to the driving shaft at times during the revolution which depend on the number standing in the first counter; for example, if a digit in the first accumulator is 9 the corresponding clutch will be let in early during the revolution, whereas if it is 1 the clutch will be let in late. All the clutches are automatically knocked out at the end of the add phase, which occupies rather more than half the rotation time of the shaft. The system will be familiar to those acquainted with Hollerith or IBM equipment.

It may be of interest to explain briefly the way in which orders are punched on the control tape. Each accumulator and each set of switches is given a serial number by which it may be referred to in the programme; this serial number is, however, not punched directly on

to the tape, but a code group obtained from a conversion table is punched instead.* Code numbers are also allocated to registers associated with the input, output, and arithmetic units. The 24 places in which holes can be punched across the tape are divided into three groups of eight. The code punched in the first group determines which accumulator or register is to be connected to the main distribution bus ready for a number to be transferred to it. The code punched in the second group similarly determines which accumulator or register is to be connected to the bus in such a way that the number standing in it will be read out. The code punched in the third group determines the operation to be performed and has also various other miscellaneous functions. The tape is punched with the aid of a keyboard perforator having three groups of eight keys; each key punches one hole on the tape and the keys in each group are numbered from 1 to 8. The code group 32, for example, would be punched by pressing the keys in the appropriate group marked 3 and 2, and two holes would be punched on the tape.

In what follows, the serial number of an accumulator will be denoted by a capital letter and the code group which corresponds by a small letter. Examples of the lines of coding which must be punched on the tape to make the machine perform certain simple operations are given below. The three columns, *out*, *in*, and *misc* refer to the three groups of holes which can be punched on the tape.

	Out	In	Misc
Take x from A and add it into B	a	b	7
Take x from A and subtract it from B	a	b	732
Clear A (i.e., reset to zero)	a	a	7

Each of the above operations takes one revolution time, that is 0·3 sec, to perform. The code 7 punched in the miscellaneous column causes the machine to proceed to the next line; if it were absent the machine would stop. The control circuits are specially wired so as to detect when the same code is punched in the out and in columns, and to cause the counter concerned to be cleared.

For more complex operations several lines of coding may be required, as, for example, the following:

Take x from A and y from B and put xy into C	a	761	—
	b	—	—
	—	c	7

The first line of coding causes the multiplicand to be delivered to the multiplier unit, which immediately forms the nine integral multiples of the multiplicand and stores them. The multiplier unit then emits a

* The code groups are so chosen that the serial number in binary form is punched on the tape; this fact is, however, of no significance to the programmer.

signal which causes the sequencing mechanism to step and read the second line of coding which causes the product to be formed; the third line is then called for and causes the product to be transferred to the correct accumulator. A multiplication takes about 6 sec, depending on the number of non-zero digits in the multiplier. Most of this time is taken up in forming the integral multiples of the multiplicand, and if the programmer wishes he may use this time to perform other operations by including extra lines of coding between the first two lines associated with the multiplication. If he does this he must punch a 7 in the miscellaneous column of the first line in order that the machine shall proceed to read the next line without waiting for the signal from the multiplier unit. He must be careful to make sure that he does not include more operations than there is time to perform. Division, which takes 11·4 sec, is called for by a somewhat similar group of three lines of coding.

The machine is provided with two punched-card feeds for input and one card punch and two typewriters for output. An example of the coding necessary for these operations is as follows:

x lies in A; print x on typewriter 1 $\left\{\begin{array}{lll} a & 7432 & — \\ — & 752 & 7 \end{array}\right.$

There are built-in mechanisms for the computation of $\sin x$, 10^x, and $\log_{10} x$. These take respectively 60, 61·2, and 68·4 sec. The mechanisms are called into action by groups of coding, of which the following is an example:

x lies in A; put $\sin x$ into B $\left\{\begin{array}{lll} a & 7631 & — \\ 84 & b & 7 \\ — & — & 7321 \end{array}\right.$

The mechanism calculates $\sin x$ to 23 decimals. If a lower degree of accuracy than this is sufficient, computation time is sometimes saved by including in the programme a group of orders for calculating $\sin x$ from a power series, instead of using the sine mechanism.

It would not be worth while to go into details about all the various devices and facilities provided on the machine. Mention may be made, however, of the *multiple in–out counter* which facilitates the storing of two numbers, of not more than 11 decimals each, in a single counter, and thus enables the effective storage capacity to be increased at the expense of speed. Pairs of counters connected so that tens-transmission can take place from the most significant end of one to the least significant end of the other are also provided in order to facilitate the use of double-length arithmetic.

When the machine was first built it contained two conditional orders of a specialized kind. One was for use in connexion with function tapes, and facilitated the choice of alternative tapes for different

ranges of the variable. The other caused the machine to stop if the number in a certain register was numerically greater than a specified quantity, and was used for verifying that two values of a quantity calculated in different ways agreed to the precision expected. No general conditional order of the type envisaged by Babbage was provided. This was an inconvenience, and the machine was later modified to include one. At the same time three additional tape-reading mechanisms were provided. The new conditional order caused control to be transferred from the main tape to any of the subsidiary tapes, or vice versa, if the number standing in a certain counter were negative. Endless tapes for carrying out repetitive cycles of operations could be placed in one or more of the tape-reading mechanisms. Transferring control from one tape to another is faster, and more convenient mechanically, than rolling back or advancing a single tape in the manner Babbage proposed for his Jacquard cards, and provides a sufficient degree of flexibility for practical purposes.

The Automatic Sequence Controlled Calculator has been running night and day with only short intermission since it was completed in 1944. Compared with later machines built in the Computation Laboratory at Harvard and elsewhere, it is slow and has only a small amount of storage capacity. Nevertheless, it has a permanent place in the history of the development of calculating machines, since it was the first automatic machine ever to be completed.

THE ENIAC

The first electronic automatic calculating machine to be built was the ENIAC (Electronic Numerical Integrator and Computer). In this machine storage of numbers and computing are done by means of electronic circuits; there are no moving parts except for the input and output mechanisms. The ENIAC was designed by J. Presper Eckert and John W. Mauchly and was built in the Moore School of Electrical Engineering, University of Pennsylvania; it was completed in the summer of 1946. The use of electronic instead of mechanical devices enables very high speeds of operation to be obtained. An addition takes 200 μsec on the ENIAC, a multiplication 2·8 msec (10 decimals by 10 decimals), and a division 6 msec (10 decimals in the quotient). It will be worth while to give some account of the electronic circuits used in the ENIAC, since more modern machines, although differing very greatly from the ENIAC, use similar basic techniques.

It is useful to discuss the design of a computing machine with the aid of highly simplified schematic diagrams. The amount of detail included in such a diagram depends on the purpose for which it is intended. For example, in discussing the 'logical design' of a machine, it is not usually necessary to include in the diagram such items as

amplifiers and pulse-shaping circuits, although these will frequently be needed in any practical machine to compensate for attenuation and distortion of the pulses.

The principal basic circuit elements used in the ENIAC are (i) *gates* or *'and' circuits* which act as electronic switches and enable the necessary connexion to be made between one part of the machine and another, (ii) bi-stable circuits, known as *flip-flops*, which are the basic units used for storage, and (iii) *'or' circuits* or *buffer circuits* which enable pulses from different sources to be combined.

A gate much used in the ENIAC is shown in Fig. 1·8. It consists of a pentode with the incoming pulses applied to the control grid and the controlling waveform to the suppressor grid. Both grids are biased negatively so that current only flows in the anode circuit when

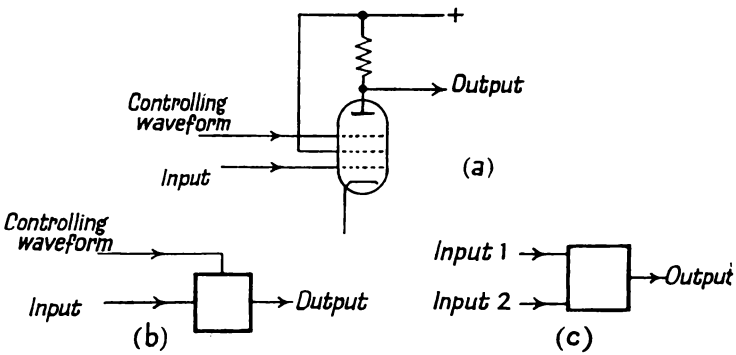


Fig. 1·8.—An electronic gate.

an incoming pulse carries the control grid positive,* and then only if the suppressor has been carried positive by the controlling waveform. When more complicated switches are required they must be built up from a number of gates. Although it is often convenient to think in terms of incoming pulses and controlling waveforms, the circuit is, from the logical point of view, symmetrical and pulses could be applied to both grids. Current will then flow in the anode circuit only at those times when pulses are applied simultaneously on both grids. From this point of view a gate may be regarded as a coincidence circuit. Fig. 1·8 also shows the schematic notation which will be used in this book to denote a gate. For some purposes it is convenient to show the two inputs unsymmetrically, as at (*b*), and for other purposes to show them symmetrically, as at (*c*).

Fig. 1·9 shows a simple form of flip-flop. The values of the resistors are chosen so that the circuit is stable when either valve is completely

* All pulses will be taken to be positive-going unless otherwise stated.

cut off (the other being conducting) and is unstable whenever both valves are passing current. It can be changed from one stable state to another by applying a pulse to the grid of the valve which is cut off.

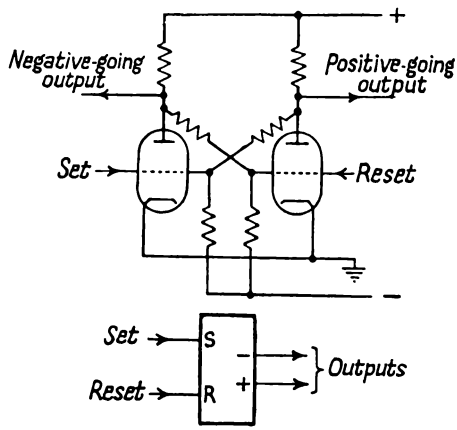


Fig. 1·9.—A flip-flop.

It is often convenient, when describing the action of a circuit containing a flip-flop, to regard one state of the flip-flop as the normal, or 'reset', condition and the other as the excited, or 'set', condition, although the choice of which state is which is arbitrary. Fig. 1·9 also

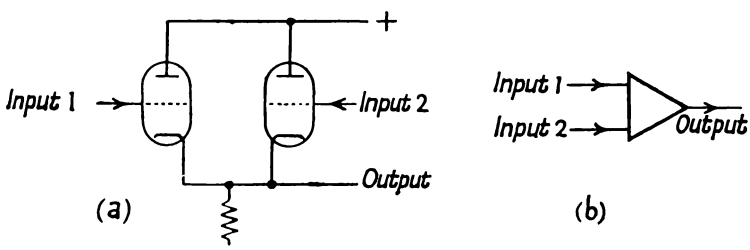


Fig. 1·10.—An 'or' circuit.

shows the schematic notation which will be used for a flip-flop. The flip-flop has two output connexions and the convention adopted is that the one marked '−' is normally at a high potential and falls when the flip-flop is set, while the one marked '+' is normally at a low potential and rises when the flip-flop is set. The flip-flops used in the ENIAC are of slightly more complicated design than that shown in

Fig. 1·9, and include buffer valves in the set and reset lines. The 'or' circuits used in the ENIAC consist of a pair of valves with a common resistor in either their anode or cathode circuits. The latter alternative is illustrated in Fig. 1·10, which also shows the schematic notation which will be adopted in this book to denote 'or' circuits. Detailed information about the circuits used in the ENIAC is given in reference 21.

Ten flip-flops may be connected in a ring so as to form a scale-of-ten counter in the way shown in Fig. 1·11. One flip-flop only is in a set condition at any given time. When a pulse is applied to the common reset line this flip-flop is reset and the output lead connected to the next stage rises in potential. This causes a positive-going pulse to be applied through a condenser to the set input of the next flip-flop in the ring. Thus every time a pulse is applied to the common input

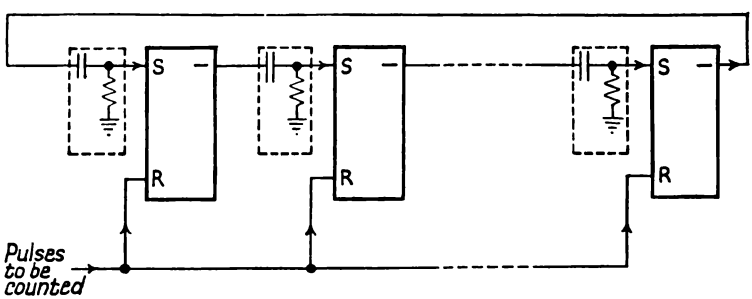


Fig. 1·11.—A scale-of-ten counter using flip-flops.

line the excitation is advanced one step. Rings of flip-flops connected in this way serve exactly the same purpose in the ENIAC as counter wheels serve in mechanical calculating machines.

The ENIAC resembles the Automatic Sequence Controlled Calculator in that numbers are stored in accumulator registers which have tens-transmission associated with them so that a number can be transferred from one accumulator and added to the number in another accumulator without passing through an arithmetic unit. There are 20 accumulators, each capable of holding a number with 10 decimal digits. The ENIAC has a multiplier and a combined divider and device for taking square roots. It has also sets of switches on which some 300 numbers may be set by hand.

The accumulator registers consist of rings of flip-flops (10 in number) provided with a system of tens-transmission similar in principle to Babbage's anticipatory carry. In addition to the 10 decade counter rings, each accumulator has a pair of flip-flops which contain the sign of the number stored in it.

The use of anticipatory carry enables a faster electronic accumulator to be built than would be possible if successive carry were used. It does not, however, as in a mechanical accumulator, render the carry time independent of the length of the register. This is because the time taken for a carry pulse to be propagated through a series of gates from one end of the register to the other is appreciably greater than the time taken for one of the counter rings to be advanced one step. The situation is thus quite different from that in a mechanical accumulator in which the counter wheels take an appreciable time to move from one position to another but in which the time taken to communicate motion, by means of a linkage, from one end of the register to the other is entirely negligible.

Numbers are conveyed from one part of the ENIAC to another by means of groups of 11 wires, one for each decimal digit and one for the sign. The number of pulses which pass along each wire is equal to the value of the corresponding digit. If these pulses are routed to the 10 counter rings of an accumulator, it is clear that the incoming number will be added to the number already in that accumulator.

The fundamental waveforms used in the ENIAC are provided by a unit known as the *cycling unit*. This contains an oscillator running at 100 kc/s from which pulses 2 μsec wide are obtained. By means of suitable counting down and gating circuits the waveforms shown in Fig. 1·12 are generated. These repeat every 200 μsec. This period is

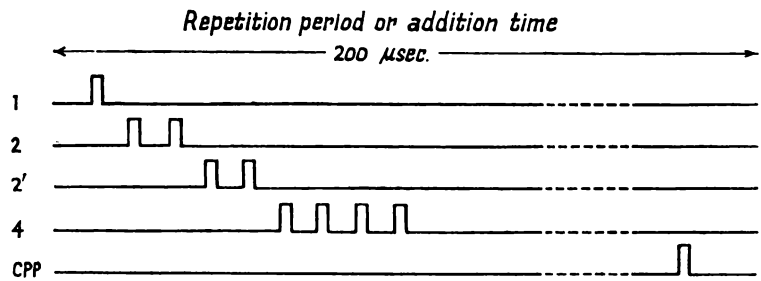


Fig. 1·12.—Fundamental waveforms used in the ENIAC.

the fundamental unit of operating time and is known as the *addition time*. Every operation takes an integral number of addition times.

The waveforms marked 1, 2, 2', and 4 in Fig. 1·12 are used to build up trains of pulses representing the digits from 0 to 9 in rather the same manner as the individual weights in a box of weights are used to build up any required weight. An example of how this is done will be given in the next paragraph when the mechanism used for the input of numbers—the *constant transmitter*—is described. The remaining

waveform shown in Fig. 1·12 is known as the *central programme pulse*. This is emitted at the end of each addition time and is used in connexion with the control circuits. In addition to the waveforms shown a number of other waveforms are generated in the cycling unit.

Numbers are fed into the ENIAC from standard 80-column IBM cards. Fig. 1·13 will help to explain how the digits punched on the

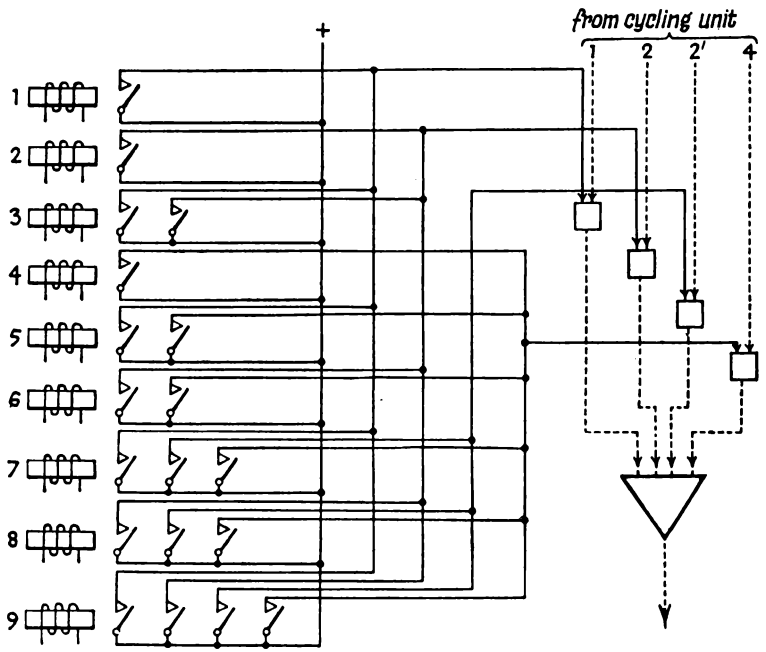


Fig. 1·13.—Circuit used in conjunction with an IBM card reader.

cards are translated into trains of pulses. The cards pass through a standard card reader, and a contact made through the punched hole in each column causes one of a set of 10 relays to be closed and to stay closed until the next card is about to be read. The principal contacts of these relays are shown in the diagram. They control a group of four gates which are fed with the 1, 2, 2′, and 4 pulses from the cycling unit. If, for example, the digit punched is 3, then the third relay from the top will be closed and the gates controlling the 1 and 2 pulses will be opened, allowing a total of three pulses to pass through in each addition time. Further gates, not shown in Fig. 1·13, are connected in such a way that the train of pulses is only emitted during the addition time for which it is required. The cards used have room

for digits to be punched in 80 columns so that eight 10-figure numbers can be transmitted at once. Punched cards are also used for output.

In addition to the property mentioned above, whereby an accumulator can receive a number transmitted in pulse form from another part of the machine and add it to the number which it already contains, an accumulator can be made to 'recycle' or emit in pulse form the number stored in it. This is achieved in the following manner. A stream of 10 pulses is passed into each ring, which is thereby taken round a complete cycle. Pulses from the ring are prevented from reaching the output line by a gate which is normally closed; when the ring passes the '9' position this gate is opened and pulses are, therefore, emitted synchronously with the incoming pulses until the end of the operation. In this way a number of pulses corresponding to the digits stored in the ring is emitted and the ring is left in the same state as it was in initially. Alternatively, the complement (see page 40) of the number standing in the accumulator can be emitted.

Control Circuits of the ENIAC

In order to make effective use of the high computing speed of the ENIAC it was necessary to provide a faster method of control than anything that could be obtained by the use of punched tape. A programme is put on the ENIAC by setting up a number of connexions through which control pulses, or *programme pulses* as they are usually called, are passed from one unit of the machine to another. These connexions are in addition to the connexions which must be made to enable pulses representing numbers to pass from one part of the machine to another. When any unit receives a programme pulse it is stimulated to perform an appropriate action; for example, an accumulator can be stimulated either to transmit to another accumulator the number contained in it, or to place itself in a state of readiness to receive a number from another accumulator. The circuit arrangements by which this is achieved will be indicated in the next paragraph. When the action called for has been completed, the unit emits another programme pulse which passes to the unit or units due to operate next. Putting a programme on the ENIAC involves making a large number of connexions with plugs and sockets and setting up a large number of switches. It is, therefore, a tedious operation. The system does, however, lead to very fast operation, since no time is needed for reading and setting up orders. It has the considerable disadvantage that changing the machine over from one problem to another is a lengthy job.

Each accumulator has five input channels through any of which it can receive digit pulses from other units. It has also two output channels through which it can transmit numbers or their complements to

other units. These input and output channels are protected by sets of gates, all normally closed, and only when the accumulator is stimulated by a programme pulse received through a separate programme input channel does any actual reception or transmission take place. There are 12 programme input channels in all, and in the case of eight of them the programmer can arrange, by setting a switch, for the operation called for to be repeated any number of times up to nine before the programme pulse indicating the end of the operation is emitted. The programme circuits are complex, but their manner of

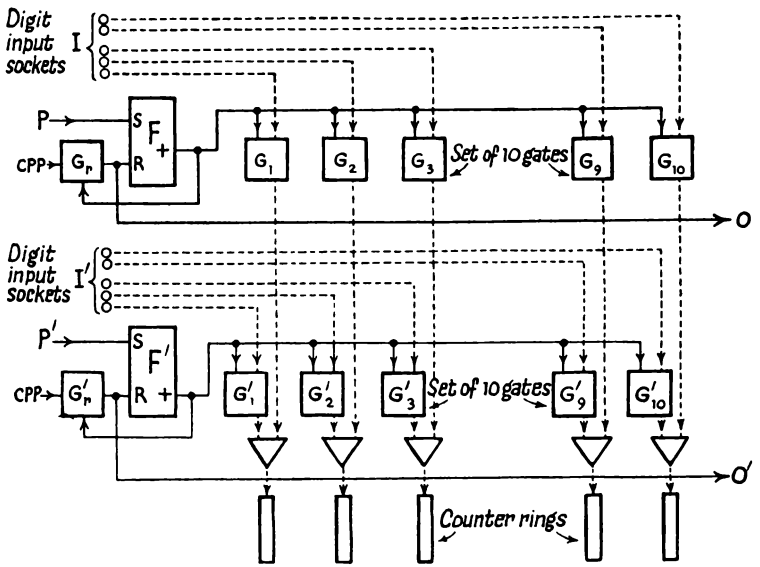


Fig. 1·14.—Simplified block diagram of the programme circuits of an accumulator.

working may be explained by reference to Fig. 1·14, which is a simplified schematic diagram showing how a programme pulse can put an accumulator into a state of readiness to receive pulses through one of the digit input channels. Two input digit channels, I and I', are shown and two programme pulse inputs, P and P'. In the normal condition the counter rings of the accumulator are isolated from the input channel I by the set of gates G_1, G_2, &c., which are controlled by the positive-going output of flip-flop F. If a programme pulse appears on the input P, the flip-flop is set and the gates are opened; any pulses appearing at the digit input I therefore pass into the counter rings. The gate G_r is also opened and the next central programme pulse is allowed to pass through to reset the flip-flop. The gates G_1, G_2,

&c., thus remain open for one addition time and for no longer. It is assumed that the response of the circuit is not so rapid that the gate G_r closes before the central programme pulse has passed through. This central programme pulse, in addition to resetting F, also passes to a programme output, O, whence it may be led to another unit to stimulate the next operation. In a similar way, if the accumulator is stimulated by a pulse on the programme input P', it receives digit pulses through the second digit input I' for one addition time and emits a programme pulse from O' on the completion of the operation.

The other numerical units of the machine are similarly provided with programme inputs through which they may be stimulated to perform their respective operations, and programme outputs through which they emit pulses when the operations are complete.

An important part of the programme equipment of the ENIAC, which has not so far been mentioned, is the *master programmer*. This contains a number of electronic circuits equivalent to multiway stepping switches, and is useful when it is desired to arrange for particular sequences of operations to be repeated specified numbers of times at certain stages in the programme. The master programmer thus provides facilities somewhat similar to those obtained by endless loops of tape on the Automatic Sequence Controlled Calculator, although the programmer is not limited to a small number of loops and can in fact make use of an elaborate arrangement of loops within loops if he so requires.

Since pulses of similar amplitude and width are used both for the transfer of numbers and as programme pulses in the ENIAC, it is very easy to arrange for a conditional operation. It is only necessary to take a connexion from one of the leads carrying numbers to a suitable programme input, for example, on the master programmer. For example, if it is desired to make an operation conditional on the sign of the number standing in a certain accumulator, a connexion must be taken from the wire which carries the sign pulse in one of the accumulator outlets. When the accumulator is stimulated to transmit through this outlet, a pulse appears on this wire only if the number in the accumulator is negative.

Fig. 1·15 shows the lay-out of an accumulator panel. The decade rings are mounted on narrow trays placed vertically side by side at the top of the unit. Each flip-flop is fitted with a neon lamp which glows only when that flip-flop is set. These lamps are visible through holes in the front panel so that when the machine is stopped it is possible to read the number standing in an accumulator. Normally, when the machine is in action, the numbers are changing much too rapidly for this to be done. Fig. 1·16 shows the lay-out of the switch panel of an accumulator. This will serve to emphasize the fact that setting up the ENIAC is a matter of some complexity.

Fig. 1·15.—Layout of an accumulator panel of the ENIAC.

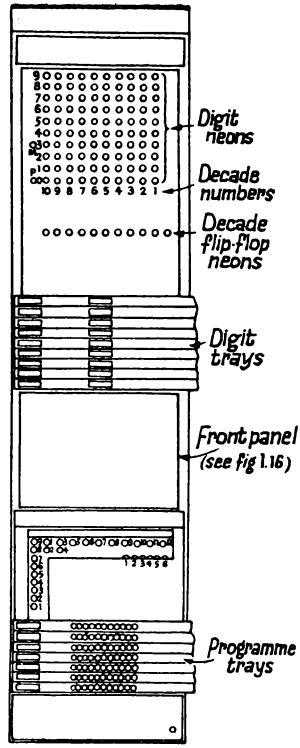
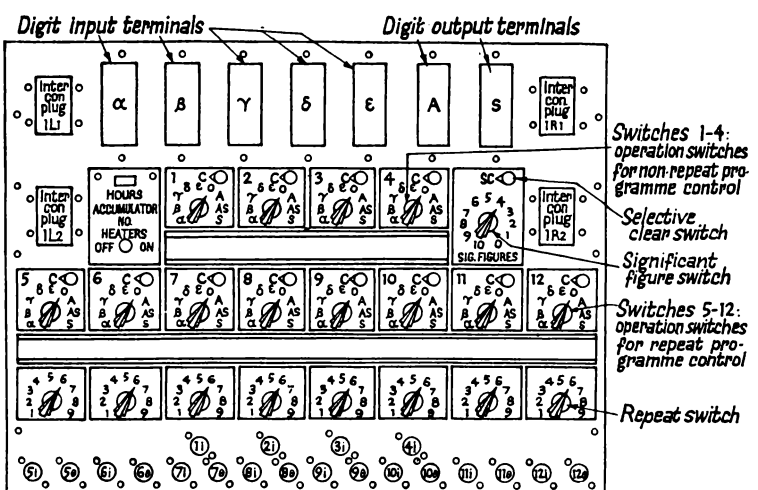


Fig. 1·16.—Layout of an accumulator switch panel.


Digit input terminals
Digit output terminals
Switches 1-4: operation switches for non-repeat programme control
Selective clear switch
Significant figure switch
Switches 5-12: operation switches for repeat programme control
Repeat switch

A general view of the ENIAC is given in Fig. 1·17. It occupies a room some 40 ft by 20 ft. It contains about 18,000 valves, and 1,500 relays, and consumes 150 kW of power. It was a *tour de force* of electronic engineering in that the designers attained their end by straightforward methods and were not deterred by the large amount of equipment necessary. As will be shown in the next section, the development of logical principles—to which the designers of the ENIAC largely contributed—has made it possible to build much more powerful machines with a good deal less equipment. The ENIAC was a successful machine, however, in spite of its great size, and this was due to the high standard of the engineering design. As a general-purpose machine its utility is limited by its small storage capacity, but it is only fair to say that when it was built its main role was intended to be the calculation of ballistic tables by the integration of an ordinary differential equation. For this purpose the storage capacity provided was adequate.

THE EDVAC

From what has been said so far it will be appreciated that an automatic calculating machine must have some device for storing numbers used during the calculation. There must also be some way in which the programme of orders can be placed in the machine before the calculation starts; in other words, provision must be made for the storage of orders. These two requirements are not as different as at first sight they might seem to be. Perhaps the best way of appreciating this is to observe that orders can be expressed in coded form as numbers; for example, we might take 1 to represent addition, 2 to represent subtraction, 3 to represent multiplication, and 4 to represent division. This would enable us to code the operation part of the order. The storage locations referred to could be represented by their serial numbers placed in a specified sequence; an example of a complete order would then be as follows:

3,118,124,002.

This would indicate that the number in storage location 118 was to be multiplied by the number in storage location 124 and that the result was to be placed in storage location 2. Once it is realized that orders and numbers may be represented in the same form, it will be seen that if so desired the same store can be used for both. A channel must be provided by which orders can pass from the store into the control, and also a channel by which numbers can pass from the store into the arithmetic unit. In order to avoid specifying whether the content of a given storage location is to be regarded as a number or an order, it is sometimes convenient to refer to it as a *word*.

One result of having the orders expressed in a coded numerical

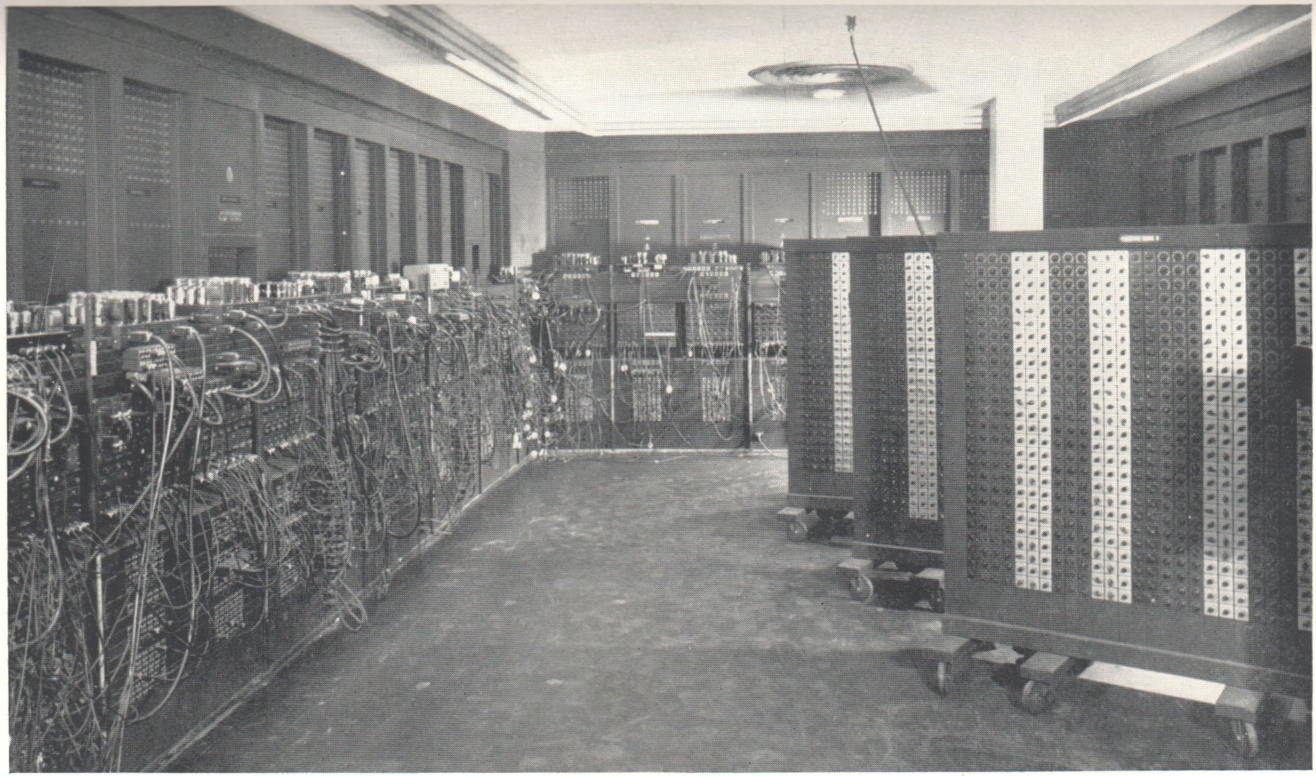


FIG. 1·17.—GENERAL VIEW OF THE ENIAC

form and held in the same store as the numbers is that it is possible to make the machine modify orders by adding constants to the numbers which represent them; for example, if the order given above (or rather the number representing it) has 1 added to it the result of the multiplication is placed in storage location 3 instead of in storage location 2. Later on, when examples are given of the way in which programmes are constructed for a machine built on these principles, it will become clear how important this feature is.

The ideas described above first saw the light of day in a report drafted by J. von Neumann and issued in June 1945, on behalf of a group associated with the Moore School of Electrical Engineering, University of Pennsylvania, where the ENIAC had just been finished. This report contained fairly specific proposals for the logical design of a machine known as the EDVAC (Electronic Discrete Variable Automatic Computer). These proposals were elaborated in later reports issued by the Moore School. The size of the EDVAC was to be only a fraction of that of the ENIAC, but it was to be of much greater power as a computing instrument.

The EDVAC was further to differ from the Harvard Machine and the ENIAC in that all numbers were to be represented (within the machine) in the binary scale, i.e., in the scale of 2. In this scale the

Fig. 1·18.—Representation of the binary number 1011 by a train of pulses.

digits are either 0 or 1, and a number can be very conveniently represented by a train of pulses in which the presence of a pulse in a particular position indicates a 1, and the absence of a pulse a 0. As an example, Fig. 1·18 shows how the binary number 1011 can be so represented. In the decimal scale this number is equivalent to $1.2^3 + 0.2^2 + 1.2 + 1$, or 11. Note that the pulse representing the least significant digit comes first in time.

The designers of the EDVAC realized that if the machine were to contain sufficient storage capacity to hold the large number of orders likely to be required in a typical programme, and at the same time be able to accommodate a large quantity of numerical information, some method of storage other than the use of flip-flops would be necessary. Their proposal was to use ultrasonic delay units such as had already been developed experimentally for other purposes. These consist of tubes filled with a liquid (in practice mercury) and closed at each end by a quartz crystal. Electrical pulses applied to one of the

crystals are converted into pulses of ultrasonic energy which travel down the column of liquid. At the far end they are reconverted to electric pulses by the other crystal. In the ultrasonic store of the EDVAC the distance travelled in the mercury is about 58 cm and the time taken 384 μsec. The pulses are derived from a *clock pulse* oscillator of frequency 1 Mc/s, so that it is possible for 384 of them to be in transit down the column of mercury at any given time. When the pulses emerge they are amplified, reshaped, synchronized, and passed back into the input of the delay unit. A train of pulses once introduced into the delay unit will continue to circulate as long as the power is switched on.

In each ultrasonic delay unit in the EDVAC eight numbers, each of 43 binary digits plus a sign digit (corresponding to about 13 decimal digits), can be stored in dynamic form as trains of pulses. There are four blank pulses between adjacent numbers. Machines in which the digits of a number follow one another in time sequence in the same channel are said to operate in the *serial* mode.

The reshaping of pulses before they are passed back to the input of the delay unit is an essential feature of the system, since, if this were not done, a progressive deterioration of form would take place

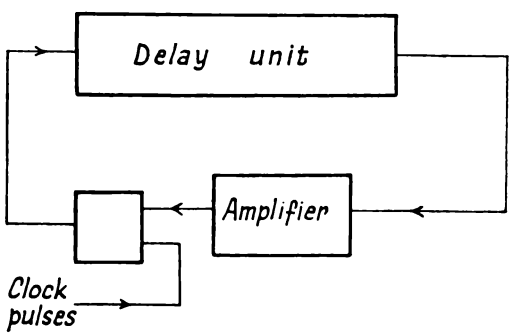


Fig. 1·19.—Regeneration and circulation of pulses in an ultrasonic delay unit.

and the pulses would be lost in a short time. Synchronizing is also necessary in order that numbers circulating in the store may remain in step with the clock and so be available when required. Both reshaping and synchronizing are achieved by means of the circuit shown in Fig. 1·19. The pulses from the amplifier are not themselves applied directly to the input of the delay unit, as was implied above, but are used to control the passage through a gate of pulses from the clock pulse generator. It is arranged that when the pulses from the amplifier arrive at the gate they are slightly wider than, and slightly in

advance of, the clock pulses. The result is that when a pulse emerges from the amplifier a cleanly gated clock pulse is allowed to pass into the delay unit. If no pulse emerges from the amplifier no pulse passes into the delay unit. This is equivalent to the continued recirculation of whatever pulses happen to be in the delay unit with reshaping and synchronizing at each passage. The diagram of Fig. 1·19 is intended only to show the principle on which regeneration takes place. Additional gates are necessary in a practical storage unit in order to permit numbers to be transferred in and out and to be cleared when no longer required.

The adoption of the binary system for the internal representation of numbers in the EDVAC was proposed because it appeared that the components out of which an electronic calculating machine can be constructed are themselves of a binary nature. This is clearly true of the ultrasonic store which has just been described, and it is also true of the bi-stable circuits, or flip-flops, which can be used for the static storage of information and for the control of switching operations. It is quite possible to construct a decimal machine out of these elements by grouping them in suitable combinations. Such a machine will, however, be a 'disguised' binary machine. In this sense the ENIAC is a disguised binary machine, whereas the Automatic Sequence Controlled Calculator, which uses counter wheels with ten positions for storage elements, is a true decimal machine. The originators of the EDVAC were the first to put forward the view, which is now held by many people, that if binary elements are to be used it is more efficient to build a true binary machine and to arrange for conversion of numbers to and from the decimal system as they go in and out of the machine. They pointed out that, since this conversion is primarily a computing operation, it can be performed by a suitable sequence of orders in the programme without the provision of a special unit in the machine.

Logical Diagrams

The early report on the EDVAC contained a detailed discussion, illustrated by schematic diagrams, of ways in which orders and numbers expressed in binary form as trains of pulses might be handled. In particular, there was a discussion of logical design of computing circuits. The schematic notation used in this report was an adaptation of one introduced by McCulloch and Pitts in connexion with the analysis of a model of the nervous system. While this notation served a useful purpose in the early stages of the development of logical design, it has the disadvantage that the symbols used in it do not always correspond directly to the circuit elements from which electronic computers are constructed. It will not therefore be used in this book. Instead use will be made of the symbolic notation already introduced in

connexion with the ENIAC augmented by the symbols shown in Fig. 1·20. These include a three-input gate which gives out a pulse if pulses are applied simultaneously to all three inputs, and a three-input 'or' circuit which gives out a pulse if a pulse is applied to at least one input. There is also an *inverting circuit* denoted by a circle containing the letter I. If this is used in conjunction with a gate, as shown in Fig. 1·20(*c*), an *inhibiting circuit* may be formed. This circuit has the property that a pulse applied to the input y will emerge from the output unless another pulse is applied simultaneously to the input x. Fig. 1·20 also includes a symbol for a delay unit capable of introducing a delay of a few pulse intervals. In practice this might consist of an

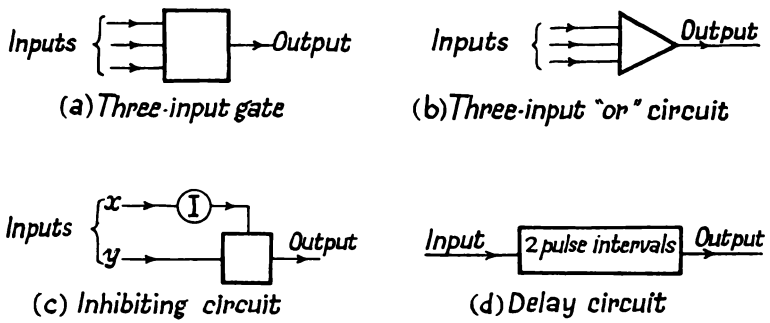


Fig. 1·20.—Symbols used in logical diagrams.

electro-magnetic delay line, although there are other circuit devices which might also be used.

It is now possible to show how a circuit may be constructed which will accept two binary numbers expressed in serial form as trains of pulses (the least significant digits coming first) and emit the sum in the same form.

Consider first the circuit shown in Fig. 1·21. This has three inputs and two outputs and, as will be seen, is capable of forming the sum and carry digits resulting from the addition of three binary digits. If

TABLE 1·2

x	y	z	S	C
0	0	0	0	0
1	0	0		
0	1	0	1	0
0	0	1		
1	1	0		
1	0	1	0	1
0	1	1		
1	1	1	1	1

the inputs are denoted by x, y, and z and the outputs by S and C, and if the presence of a pulse on a given line is indicated by a 1 and the absence of a pulse by a 0, the values which S and C take for the

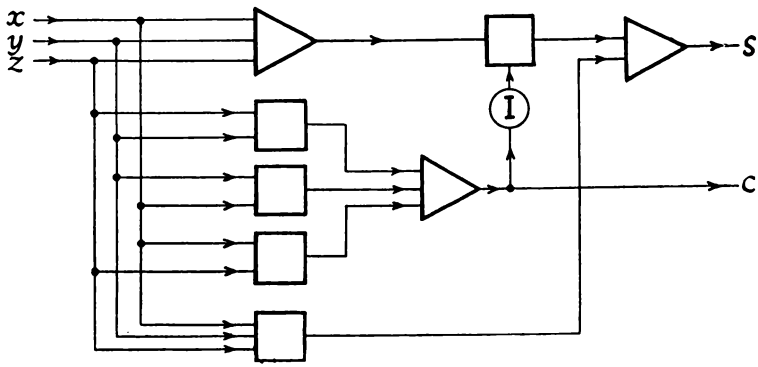


Fig. 1·21.—A three-input binary adder.

various possible combinations of values of x, y, and z are shown in Table 1·2. It may easily be verified that S and C correctly represent the sum and carry digits formed by the addition of x, y, and z.

Fig. 1·22 shows how the circuit of Fig. 1·21 may be connected to form an adding circuit suitable for use in a serial machine such as the

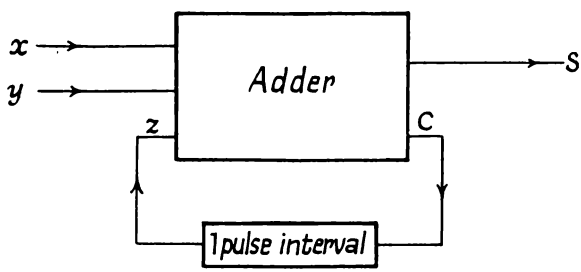


Fig. 1·22.—A serial binary adding circuit.

EDVAC. Two of the inputs are connected to lines carrying the numbers to be added, while the third is connected to the carry output through a delay unit which introduces a delay equal to one pulse interval. A carry pulse thus arrives at the input to the adder at the same time as the succeeding digits of the two numbers being added.

There are a variety of other circuits, similar in principle to that shown in Fig. 1·21 but differing in detail, which may be used for

binary addition. Further discussion of this subject will, however, be left for Chapter 6.

It has been explained that in the EDVAC numbers are represented serially, i.e., the pulses representing the individual digits of a number follow one another in time sequence. The same principle is applied to the execution of orders. Only one order is executed at a time, there being no provision for the simultaneous performance of a number of separate additions and multiplications. Not only does this enable the amount of equipment in the machine to be kept to an absolute minimum, but it greatly simplifies programming, since the programmer does not have to worry about how long the various operations take. The pulse repetition rate is much higher in the EDVAC than in the ENIAC, so that the intrinsic speed of the circuits is greater. This increase of speed is, however, balanced by the fact that all operations are performed serially and the time taken to perform (say) a multiplication on the two machines is very similar.

The ultrasonic store of the EDVAC has a capacity for 1,024 numbers. There is also an auxiliary store which has capacity for some 20,000 numbers but with a very much longer access time. This auxiliary store consists of reels of nickel-plated bronze wire on which the digits are recorded magnetically. Numbers and orders are usually transferred from one store to the other in blocks of 50 or 100 at a time. They may be recorded on a wire by means of a special piece of equipment separate from the machine, which is fitted with a keyboard similar to that on a typewriter. The wire can later be placed on one of the drive mechanisms of the EDVAC, and this constitutes the normal method of putting data into the machine. Similarly, a wire taken from the machine may be placed on another special device and the data on it transcribed in printed form. In view of later developments it is interesting to note that the need for an auxiliary store was foreseen in 1945 and that the provision of magnetic wires was an essential feature of the original proposals.

A few examples will now be given of the orders available for use by the programmer. It is convenient to refer to the serial number of the storage location which contains a given order or number as the *address* of that order or number; conversely, the content of a given storage location n is denoted by $C(n)$. Thus n is the address of $C(n)$. In the example given on p. 30 (showing how an order may be coded as a number) the order contains reference to three locations in the store, that is, it contains three addresses. An order code in which the orders are of this type is known as a *three-address code*. It is implied that the orders are normally taken from the store and executed in consecutive order. In the EDVAC order code, however, there is a fourth address which specifies the location from which the next order is to be taken. For convenience of mental association, when orders

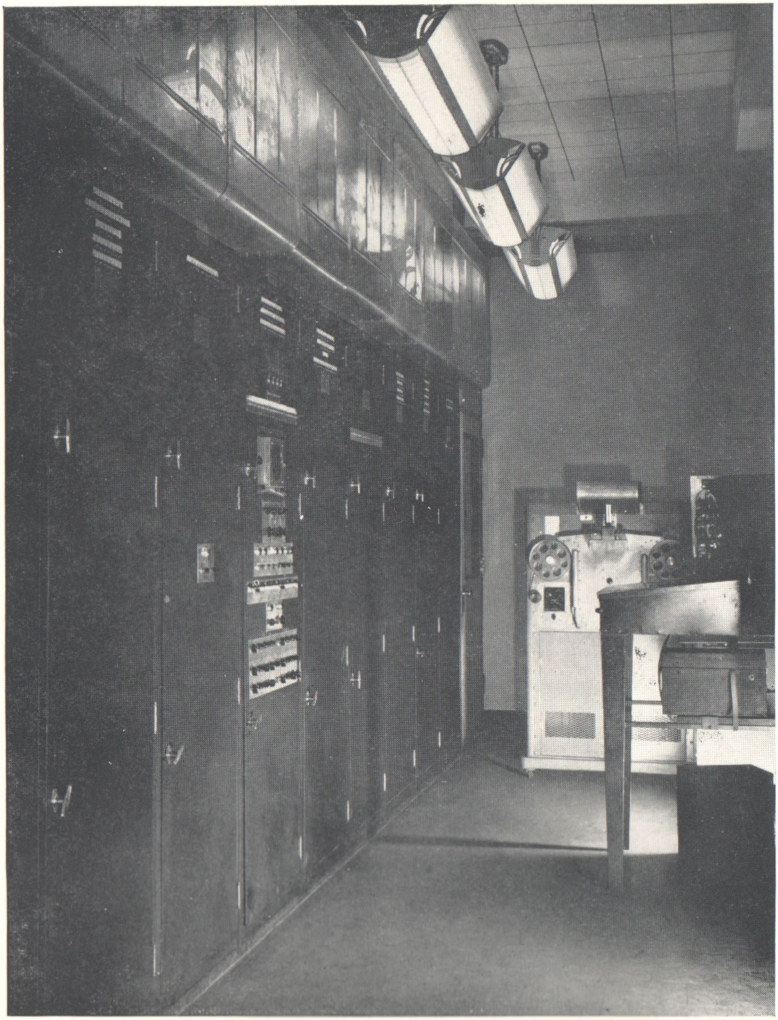


FIG. 1·23.—GENERAL VIEW OF THE EDVAC

are written outside the machine the function part of the order is represented by a letter, e.g., A for add, S for subtract. These letters are translated into corresponding binary numbers according to a fixed code when the orders are taken into the store of the machine.

Examples of arithmetic orders are as follows. In each case δ is the address of the next order.

	Address no. 1	Address no. 2	Address no. 3	Address no. 4	Operation
Form $C(\alpha)+C(\beta)$ and place in γ	α	β	γ	δ	A
Form $C(\alpha)-C(\beta)$ and place in γ	α	β	γ	δ	S
Multiply $C(\alpha)$ by $C(\beta)$ and place result in γ (rounded-off to 43 binary places) .	α	β	γ	δ	M
Divide $C(\alpha)$ by $C(\beta)$ and place result in γ (rounded-off to 43 binary places) . .	α	β	γ	δ	D

A conditional order is as follows:

If $C(\alpha) \geq C(\beta)$ take the next order from δ; otherwise take the next from δ' .	α	β	δ'	δ	C

In orders which control the reading and writing of information on the magnetic wires the second address is split into two parts. The first part helps to define the operation to be performed, and is to be associated logically with the operation part of the order. The second part gives the number of the wire concerned. Examples of wire control orders are as follows:

Write on wire n in sequence the contents of storage locations $\alpha, \alpha+1, \alpha+2, \ldots\ldots, \gamma$.	α	$1, n$	γ	δ	W
Read from wire n a sequence of numbers and place them in storage locations $\alpha, \alpha+1, \alpha+2, \ldots\ldots, \gamma$ . .	α	$2, n$	γ	δ	W

Although the mathematical requirements for a high-speed automatic computing machine were clearly understood in 1945 and 1946 and although nothing was proposed which was not apparently well within the technical resources available at the time, the construction of the EDVAC and of similar machines that were started shortly afterwards took much longer than experience in building the ENIAC had led people to expect. This is no doubt partly due to the fact that the new machines used much faster circuits than the ENIAC. Not only is the pulse repetition frequency much higher but the duty cycle is also greater, that is, the pulse trains carrying information succeed each other with much smaller quiescent intervals than in the ENIAC. One might say that the information carrying circuits in the EDVAC

are used more efficiently than those in the ENIAC and that this tends to reduce the margins for satisfactory operation unless corresponding improvements in electronic technique are made. At all events the various groups working on high-speed machines found it more difficult to make their machines work satisfactorily than they had expected, and partly as a result of this the first machines to come into operation were those planned on comparatively unambitious lines, both as regards speed of operation and storage capacity provided. These machines also did not have large-capacity auxiliary stores.

A photograph of the EDVAC showing the magnetic wires is reproduced in Fig. 1·23. The EDVAC is now in use at the U.S. Army Proving Ground at Aberdeen, Md.

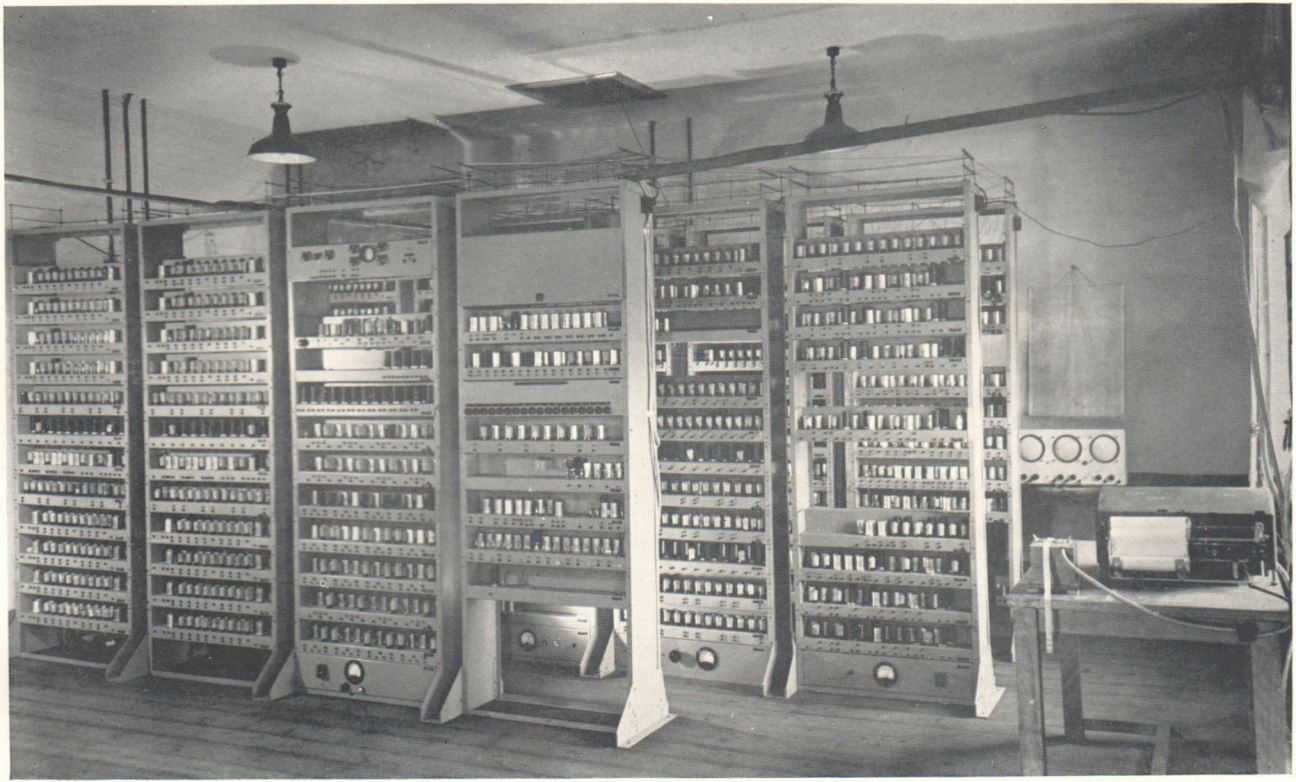


FIG. 2·1.—A PHOTOGRAPH OF THE EDSAC TAKEN SHORTLY AFTER ITS COMPLETION

CHAPTER 2

THE PRINCIPLES OF LOGICAL DESIGN

MODERN electronic calculating machines differ to a considerable extent in logical design, although the general principles on which the logical design is based are the same in all cases. The object of this chapter is to acquaint the reader with these principles and with some of the different kinds of machines which can be based on them. The method adopted will be to describe first a particular machine in some detail and then to discuss in more general terms the various alternatives which confront a designer when he is considering what logical design to adopt for a new machine. This discussion will be illustrated by references to various automatic calculating machines now in operation, with special reference to those which have influenced thinking on logical design. No detailed description of any of these machines will, however, be attempted.

The machine chosen for detailed logical description is the EDSAC, a general view of which is given in Fig. 2·1. This machine was designed and constructed in the University Mathematical Laboratory at Cambridge, and I am therefore better acquainted with its details than with those of other machines. It has, moreover, a straightforward logical design and is fairly typical of serial machines with a single-address order code. Design of the EDSAC was begun at the end of 1946, and the machine performed its first completely automatic calculation in May 1949.

The discussion of logical design cannot proceed very far without some reference to the representation and handling of negative numbers. A discussion of the various methods available for doing this will therefore be given before the main subject-matter of the chapter is entered upon.

NEGATIVE NUMBERS

In ordinary arithmetic a negative number is usually written as an essentially positive number preceded by a minus sign. This method can be used in automatic computing machines; for example, in a decimal machine the registers can consist of a row of counter wheels for storing the modulus of the number and a separate wheel capable of taking up two distinct positions for storing the sign. An objection to this system, however, is that it makes addition and subtraction complicated; for example, when two numbers have to be added together the machine must first examine the signs of the two numbers, and if these are the same it must add the two moduli, otherwise it must

subtract them. In the latter case the smaller modulus must be subtracted from the larger, and not vice versa, and the correct sign must be appended to the result.

These difficulties with addition and subtraction are avoided completely if negative numbers are represented by the complements of their moduli. The complement of a number p with respect to any constant A may be defined as $A-p$. For the present purpose A is taken to be the number which is just too large to be stored in a register; thus, in a decimal machine $-p$ would be represented by 10^r-p, where 10^r-1 is the largest number which can be stored in the accumulator register. 10^r-p is the number which would be obtained in the accumulator register if p were subtracted from 0 and an overflow allowed to occur at the left-hand end. In order that negative numbers may be distinguished from positive numbers it is necessary to restrict the range in which a number x may lie, for example, by requiring that $-10^{r-1} \leqslant x < 10^{r-1}$. If this is done the leading digit is a 9 in the case of a negative number and a 0 in the case of a positive number; this digit is usually referred to as the *sign digit*. Representation of negative numbers by complements is familiar in the handling of logarithms of numbers less than 1; for example, $\log_{10} \frac{1}{2}$ is often written as 9·6990 instead of $-0{\cdot}3010$.

If a negative number represented in complementary form is divided by 10 an extra 9 must be introduced; for example, 9·6990 becomes 9·96990. This is easily seen to be so if the 9 appearing as a sign digit is regarded as standing for a series of 9s stretching indefinitely to the left. It will also be observed that the sign digit 9 is equivalent to -1, and it is sometimes useful to treat it in this way inside a computing machine. Doing this corresponds to writing $\log_{10} \frac{1}{2}$ as $\bar{1}{\cdot}6990$.

Another way of describing the complement used to represent $-p$ is to say that it is the result of increasing $-p$ by 10^r. Since a positive number is unaltered, as far as a register of capacity 10^r-1 is concerned, by having 10^r added to it, the use of complements to represent negative numbers is equivalent to adopting the convention that *all* numbers are to be increased by 10^r, thus avoiding the occurrence of negative numbers altogether.

The remarks of the last two paragraphs apply equally well to numbers expressed in the binary scale if powers of 10 are replaced by powers of 2. The sign digit is then 0 in the case of a positive number and 1 in the case of a negative number; the sign digit thus resembles other digits in that it takes two values only and, as far as addition and subtraction are concerned, it plays the same role as the other digits.

If d is a decimal digit, it is convenient to refer to $9-d$ as the *complement on 9* of d; similarly $10-d$ is the *complement on 10* of d. By examining a few examples the reader will see that the complement of a number (expressed in the scale of 10) may be obtained by replacing

each digit individually by its complement on 9, from the left up to the least significant non-zero digit, which is replaced by its complement on 10. If all the digits are replaced by their complements on 9, the result is called the *9s complement* of the number. 9s complements can be converted to true complements by adding 1. They are very easily formed in a calculating machine, since each digit in a number can be treated separately without reference to the other digits. For this reason 9s complements are sometimes used for the representation of negative numbers. If this is done, addition and subtraction may be carried out in the ordinary way, except that it will be found that whenever the overflow of a digit occurs on the left of an accumulator register a 1 must be added in on the right. This can be achieved automatically by connecting the carry output from the most significant digit of the register to the carry input of the least significant digit. This principle of *end-around carry*, which had long been used in connexion with punched-card machinery, was adopted for the Automatic Sequence Controlled Calculator and has since been adopted for later machines built at Harvard and elsewhere. One objection to the scheme, which has otherwise much to commend it, is that there are two forms for the representation of zero—a row of 0s or a row of 9s. It is important that the design of the machine should be such that the programmer does not need to distinguish between these two forms of zero when using conditional operations. A similar remark applies to the use of signs and moduli, since, in this system of representation, there are also two forms of zero, namely $+0$ and -0.

It may be noted that if an accumulator is fitted with end-around carry, and if overflow takes place at the left-hand end when a series of numbers are added in, the number finally standing in the accumulator is the sum, modulo $10^n - 1$, of the numbers added in (n is the number of decimal places in the accumulator). If end-around carry is not fitted, the sum is formed modulo 10^n.

In order to illustrate the three systems which may be used for the representation of negative numbers, the same operation, namely the addition of two numbers, one positive and one negative, is shown below as it appears in the three systems.

Sign and modulus	True complements	9s complements with end-around carry
38421	038421	038421
− 7528	992472	992471
30893	1 030893	030892
		→ 1
		030893

End-around carry can also be used when numbers are expressed in the binary scale and similar considerations apply.

Although the use of complements, of either type, removes the difficulty with addition and subtraction which was referred to at the beginning of this section, it has the effect of making multiplication rather more complicated than if signs and moduli are used. In the latter case it is only necessary to multiply the moduli together and to prefix a positive or negative sign according as the two numbers being multiplied are of the same or different signs. In the former case various methods are possible. One is to convert the numbers to signed form when they enter the multiplier and to convert the product, if it is negative, to complementary form when it is transferred to another part of the machine. A second method is to perform the multiplication as though both numbers were positive and to make an appropriate correction afterwards in cases where one or both of the factors is negative. If true complements are used, the correction always consists in subtracting the factor, or factors, concerned multiplied by a power of 10 (or in the case of a binary machine by a power of 2). A third method, applicable if true complements are used, is to treat the sign digits as -1. Other methods, or combinations of these three methods, are also possible.

THE EDSAC

The EDSAC resembles the EDVAC in that it is a serial automatic calculating machine working in the binary system and having an ultrasonic store. It differs from the EDVAC in that it uses a *single-address* order code, that is, each order refers to no more than one location in the store. For such an order code to be possible the arithmetic unit of the machine must contain an accumulator register in which sums of products may be held over from one order to another. As an example, an order in a single-address order code might call for a number from a specified location in the store to be added to the number standing in the accumulator, or it might call for the number standing in the accumulator to be transferred to a specified place in the store. In the EDSAC punched paper tape, of the kind used in teleprinter operation, is used for input and output.

A block diagram of the whole machine is given in Fig. 2·2, and it will be seen that connexions exist through which orders may pass from the store to the control and numbers from the store to the arithmetic unit or vice versa. In the ordinary way orders pass from the store into the control for execution in consecutive order; that is, if one order comes from storage location n, the order for the next operation will come from storage location $n+1$, the next from $n+2$, and so on. This corresponds to the consecutive reading of the cards strung over the Jacquard mechanism in Babbage's machine. If a conditional order is encountered one of two things can happen. If the condition is not satisfied, the machine goes on to take the order in the storage

location next in sequence as though the conditional order had not been present. If the condition is satisfied, then the next order is taken from some other storage location specified in the conditional order. Once control has been transferred, as we say, to the new place in the store, the machine continues to take orders consecutively, starting from there, until another conditional order is encountered. The effect

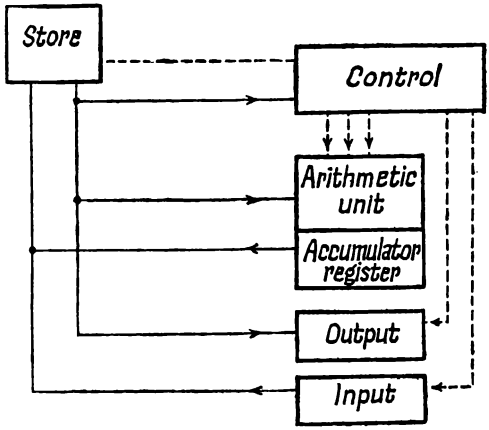


Fig. 2·2.—Block diagram of the EDSAC.

is, therefore, exactly the same as that obtained by rolling back or advancing the cards in the Analytical Engine. An order is also provided for unconditional transfer of control.

The following is a selection from the order code of the EDSAC showing the form in which orders are written and punched on the input tape. The letter F is a symbol used to indicate the end of the order. C(*n*) denotes the content of storage location *n*.

A *n* F Add C(*n*) into accumulator.
S *n* F Subtract C(*n*) from accumulator.
H *n* F Transfer C(*n*) to multiplier register.
V *n* F Multiply C(*n*) by number in multiplier register and add result into accumulator.
T *n* F Transfer number in accumulator to storage location *n* (to replace what is already there) and clear accumulator.
G *n* F If number in accumulator is negative, execute next the order in storage location *n*; otherwise proceed serially (conditional order).
E *n* F If number in accumulator is positive or zero, execute next the order in storage location *n*; otherwise proceed serially (conditional order).
F *n* F Execute next the order in storage location *n*.

There are also right- and left-shifting orders which cause the number in the accumulator to be shifted a specified number of binary places to the right or to the left, that is, to be divided or multiplied by a specified power of 2. It will be noted that two conditional orders are given, with opposite conditions for transfer of control. One such order

is all that is logically necessary, but the provision of two is justified on the ground that it serves the convenience of the programmer.

The EDSAC has no order for division, and this operation is in practice effected by a *subroutine*, or self-contained sequence of other orders. Further discussion of this subject will be found on pp. 110–12.

All numbers inside the EDSAC are regarded as lying in the range $-1 \leqslant x < 1$, and the multiplier in the arithmetic unit is wired accordingly. This is equivalent to assuming that the binary point is located immediately after the most significant digit, which is regarded as a sign digit. If it is desired to adopt some other convention it is necessary to include an appropriate right- or left-shift order after each multiplication. It is usually preferable, however, to adhere to the convention that numbers are in the range given above and to adjust all formulae and equations so that it is only necessary to handle in the machine numbers which fall within this range. This can always be done by introducing appropriate integral powers of 2.

The numerical code by which orders are represented in the store of the machine will now be given. Each function letter—A, S, &c.—has a numerical value which will be denoted by the corresponding small letter—*a*, *s*, &c. These numerical values are all integers in the range $-16 \leqslant x < 16$. Within the machine, the order A *n* F, to take an example, is represented by the number $2^{-4}a + 2^{-15}n$. Another way of expressing this is to say that the five most significant (binary) digits in an order represent the function letter, and the next 11 digits represent the address expressed as a binary integer. Since, however, the store of the EDSAC contains 1,024 locations only (numbered from 0 to 1,023), the digit immediately following the function digit is not needed to specify any address.*

The storage locations in the EDSAC can each hold a number consisting of 16 binary digits plus a sign digit, that is 17 digits in all. It is, however, possible for the programmer to combine together any two locations, provided that the first bears an even serial number, into one long location capable of holding a number with 34 binary digits plus a sign digit. (The number is 34 and not 33, since a blank digit is always left between adjacent numbers in the ultrasonic store, and this can be utilized when two short locations are run together.) At any given time the store may contain a mixture of long numbers, short numbers, and orders, orders having the same length as short numbers.

An order is made to refer to a long location by terminating the address with D instead of F; for example, the order A 100 D causes the content of the long location formed by combining the short locations 100 and 101 to be added into the accumulator. Inside the machine the order A *n* D is represented by the number $2^{-4}a + 2^{-15}n + 2^{-16}$,

* This digit was originally spare, but has recently been brought into use to control a B register (see p. 72) now fitted to the EDSAC.

which differs from the number representing A n F in that the least significant binary digit is a 1 instead of a 0. The addition of this digit is performed automatically, in a way which will be described later, when the programme tape is being read.

Examples of the binary form assumed by orders inside the EDSAC are as follows (note $a = -4$, $t = 5$):

A 20 F: $-4.2^{-4} + 20.2^{-15} = 11100000000101000$
T 520 D: $5.2^{-4} + 520.2^{-15} + 2^{-16} = 00101010000010001$

As a simple example of the use of the above order code, suppose that the numbers standing in storage locations 50 and 51 have to be added together and the sum put in storage location 100. The following orders are necessary, it being assumed that the accumulator contains zero to begin with:

A 50 F
A 51 F
T 100 F

A slightly more complicated example is the evaluation of the expression $k(x + yz)$, where k, x, y, and z are given numbers. It is supposed that $C(1) = k$, $C(2) = x$, $C(3) = y$, and $C(4) = z$. The orders required are as follows:

A 2 F
H 3 F
V 4 F
T 5 F
H 1 F
V 5 F

These orders leave the required quantity in the accumulator.

The shifting orders will be denoted here by:

R n F Shift the number in the accumulator n places to the right; i.e., multiply by 2^{-n}.
L n F Shift the number in the accumulator n places to the left; i.e., multiply by 2^{n}.

These shifting orders are slightly simpler than those which appear in the EDSAC order code, which was designed to make the construction of the shifting unit as simple as possible.

The following orders show how to evaluate the expression $2(\frac{1}{4}yz - x)$, where x, y, z are contained in the same locations as in the previous example.

H 3 F
V 4 F
R 2 F
S 2 F
L 1 F

As before, the result is left in the accumulator.

A simple example of the use of a conditional order is provided by the following programme for replacing the number in storage location 0 by its modulus:

Location of order in store	Order
10	S 0 F
11	G 13 F
12	T 0 F
13	T 1 F

The purpose of the last order is to clear the accumulator, storage location 1 being used as a 'rubbish bin'. Note that it is here necessary to specify the locations in the store into which the orders are to be placed, since the address, 13, in the second order depends on this information.

Apart from its use in arithmetic contexts, such as the foregoing, a conditional order provides a means whereby a sequence of orders may be executed as many times as desired. It thus provides the programmer with a device similar to the use of endless loops of tapes in machines like the Automatic Sequence Controlled Calculator at Harvard. The following example shows how the machine can be made to repeat a sequence of orders five times. It is supposed that $C(1) = 5.2^{-15}$ and $C(2) = 2^{-15}$.

10	S 1 F	
11	T 0 F	
12		} orders (p in number) to be repeated (these leave accumulator clear)
.		
.		
.		
$12+p$	A 0 F	
$13+p$	A 2 F	
$14+p$	G 11 F	

If the action of these orders is examined it will be seen that storage location 0 is used as a counter, the number in it starting as -5.2^{-15} and increasing by 2^{-15} at each repetition until it finally becomes zero.

Often it is desired to repeat the sequence of orders but with a slight modification each time. Here use is made of the fact that the orders are stored in coded numerical form and can therefore be modified by having an appropriate constant added on to them. For example, the order A 50 F can be turned into the order A 51 F by the addition of 2^{-15}. The following programme for adding up a sequence of 100 numbers, placed in advance in consecutive locations in the store from 50 onwards, is an example of the use of this device. The sum is placed in storage location 200, which location is assumed initially to contain zero. It will be seen that use is made of the method of counting just

described in order that the sequence of orders may be repeated the right number of times.

$C(1) = 100.2^{-15}$; $C(2) = 2^{-15}$

10	S 1 F	part of counting sequence
11	T 0 F	
12	A 200 F	add next number to partial sum in 200
13	A 50 F	
14	T 200 F	
15	A 13 F	modify previous sequence
16	A 2 F	
17	T 13 F	
18	A 0 F	part of counting sequence
19	A 2 F	
20	G 11 F	
21	. . .	

The function digits of an order, as well as the address digits, can be modified by adding a suitable constant. For example, the order A 20 F, whose numerical representation is given on p. 45, can be converted into T 520 D, whose numerical representation is also given, by adding the number $9.2^{-4} + 500.2^{-15} + 2^{-16}$.

The foregoing examples are intended to introduce the reader to the elementary principles of programming with a single-address order code. The subject will be taken up again and treated in more detail in the next chapter.

The Logical Design of the EDSAC

The high-speed store of the EDSAC consists of 32 mercury tanks each giving a delay of 576 (32×18) pulse intervals and therefore capable of holding 32 numbers of 17 binary digits (one being a sign digit) with a space equal to one pulse interval between them. A tank thus provides 32 storage locations which are numbered serially from 0 to 31. For convenience of mechanical construction the tanks are built in batteries containing 16 tanks each; there are two such batteries, and they are contained in a thermostatically controlled oven. Some details of the construction of the batteries are given in Chapter 5.

The long tanks just described form the main store of the EDSAC; in addition there are a number of short tanks, each of which holds one short or long number only. Several of these are used in the arithmetic unit to hold the numbers being operated on, and the rest are used for various purposes in other parts of the machine.

The time interval corresponding to 36 binary digits is known as a *minor cycle*, and the time taken for a complete circulation of the numbers contained in a long tank a *major cycle*. These terms were first introduced in von Neumann's original report on the EDVAC. The clock pulse interval is about 1·9 μsec so that the duration of a major cycle, which includes 16 minor cycles, is 1·1 msec. The clock pulse

generator includes a counting-down circuit which produces pulses of the same shape and duration as clock pulses but occurring only at the beginning of each minor cycle. These pulses are denoted by D_0. Similar pulses occurring one, two, three, &c., pulse intervals later are also formed for use in various parts of the machine. These are denoted by $D_1, D_2, D_3, \ldots, D_{35}$. D_{35} is coincident with the sign digit of a long number circulating in one of the tanks of the main store, D_{34} is coincident with the most significant digit, D_{33} with the next most significant digit, and so on.

A short tank, giving a delay of half a minor cycle and connected in series with an adding circuit, is used to count minor cycles. Unity is added to the number stored in this tank—called the *counter tank*—at every circulation.

There are two separate systems of interconnexions between the

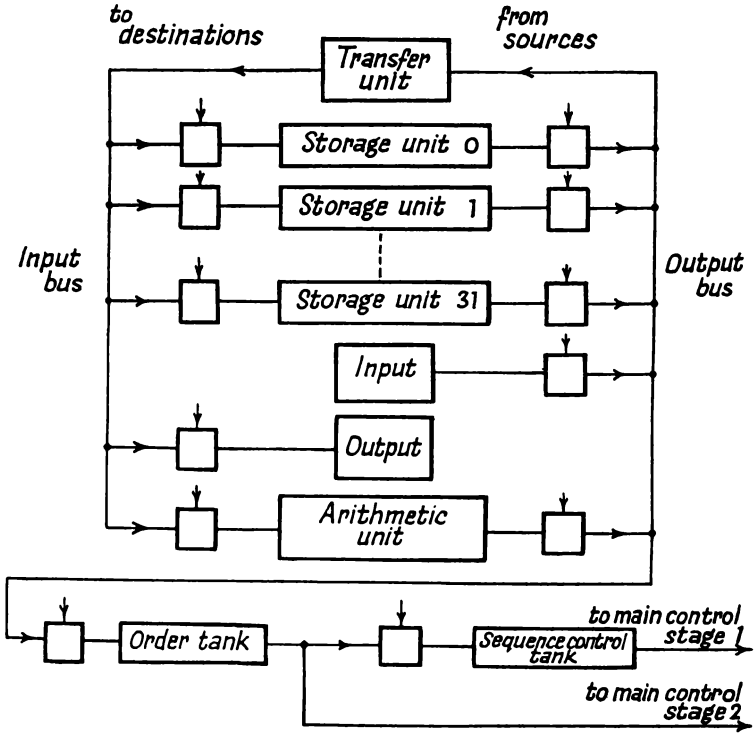


Fig. 2·3.—Numerical interconnexions in the EDSAC.

units, those for numbers and those for control waveforms. The first of these is shown in Fig. 2·3. The outputs from each tank, from the

arithmetic unit, and from the input unit are connected together to form the *output bus*. The inputs to each tank, to the arithmetic unit, and to the output unit are connected together to form the *input bus*. There is also a connexion from the output bus to the *order tank* and from the order tank to the *sequence control tank*. The purpose of these connexions will be discussed later.

The arithmetic unit treats short numbers as though they were long numbers in which the eighteen least significant digits were zeros. When a short number standing in an even-numbered storage location is transferred to the arithmetic unit, the pulses representing it must be delayed by half a minor cycle in order that the position of the binary point shall be correct. Conversely, when a short number is transferred from the arithmetic unit to an even storage location a similar delay must be introduced. The output bus is therefore connected to the input bus through the *transfer unit*, which introduces the delay when required.

The various units which are connected through gates to the output bus may be referred to as *sources* and the units which are connected through gates to the input bus as *destinations*. The execution of an order usually involves the transfer of a number from a source, through the transfer unit, to a destination, although certain orders such as shift orders concern the arithmetic unit only and do not involve such a transfer.

It has already been explained that, in normal operation, orders are taken successively from consecutive locations in the store, each order being executed before the next is taken. At the beginning of each operation the sequence control tank contains the address of the order due to be executed next. Before the completion of the operation unity is added to the address contained in this tank so as to prepare for the execution of the next order.

The Control System of the EDSAC

The action of the machine is controlled by the main control unit. It is this unit which is stimulated when the start button is pressed and which emits the waveforms which initiate the two stages into which the operating sequence of the machine falls. In Stage I an order is transferred from the storage location given by the number in the sequence control tank to the order tank, which is a short tank capable of holding a single order. In Stage II the operation called for by the order now in the order tank is executed. The beginning of Stage II is perhaps the best starting point for a detailed explanation of the control system, a block diagram of which is shown in Fig. 2·4. It will be assumed to begin with that the order in the order tank is one which calls for the transfer of a short number to or from the store. The pulses which represent the digits of the order will be denoted by

0_1 to 0_{17}. These pulses pass through the gate *G*, which, during Stage II, is opened by a waveform supplied by the main control unit. The pulses representing the function digits of the order, namely 0_{13} to 0_{17}, are used to set up a series of flip-flops in what is known as a *static register*. The digits are then available as steady voltages, lasting until the static register is reset, instead of as transient pulses. The information is still, however, in binary form. The static register is con-

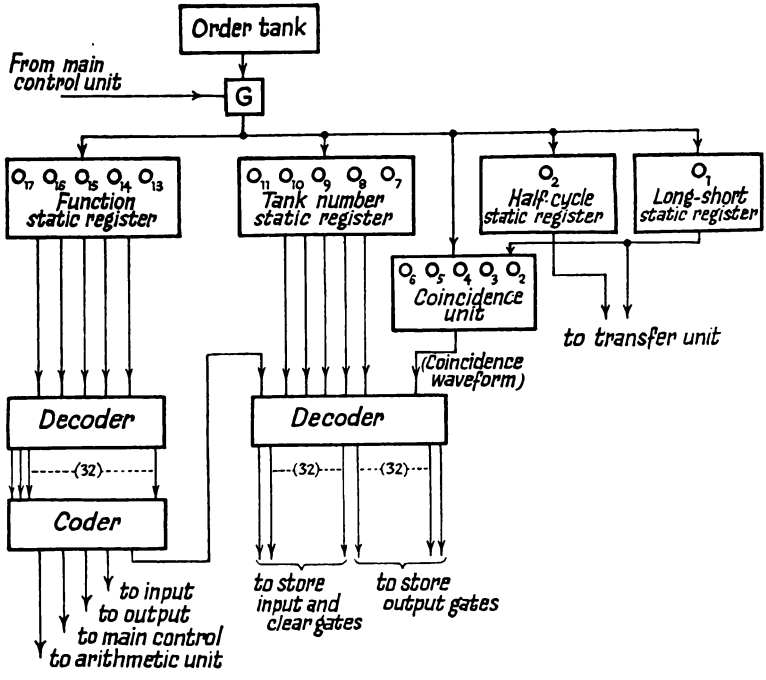


Fig. 2·4.—Block diagram showing how orders are interpreted in Stage II of the control sequence.

nected to a *decoder*, or *tree*, which has 32 outputs, of which one—and only one—rises in potential for each combination of the five digits, that is, for each type of order. Since the EDSAC has fewer than 32 orders, some of the outputs of the decoder are not used.

In order to generate output waveforms suitable for operating gates in the arithmetic unit and elsewhere in the machine, it is necessary to recode in another way the outputs of the decoder. For example, certain gates must be opened by both add and subtract orders, and for this purpose the A and S outputs of the decoder must be combined. This operation is performed in the *coder*. The static register and the

decoder perform simple logical operations and can be designed independently of the rest of the machine, but the coder must be specially designed to suit the requirements (for example) of the particular arithmetic unit in use. The coder outputs take up their potentials as soon as the static register is set and retain them until it is reset.

Of the pulses representing the address digits, 0_7 to 0_{11} specify the tank which contains the storage location referred to, and 0_2 to 0_6 the place which that storage location occupies in the tank. 0_2 to 0_6, therefore, determine the time, relative to the clock pulses, at which the required number appears at the output gate of the tank (if a transfer out of the store is to take place), or the time at which the input gate must be opened (if a transfer into the store is to take place). 0_2 to 0_6 are used to generate a waveform which opens the input or output gate of the tank and holds it open long enough for the number to pass through. The way in which this waveform is generated will be described next, and the manner in which it is switched to the input or output gate of the right tank will be described later.

It will be remembered that the number in the counter tank increases by one every half minor cycle. Each time the number, whose place in

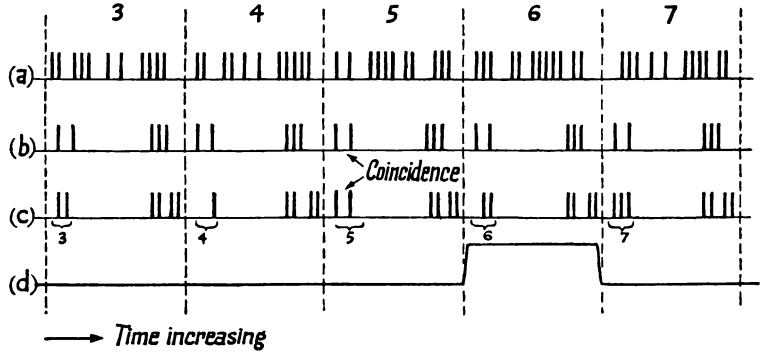


Fig. 2·5.—Diagram to illustrate the action of the coincidence unit.

(*a*) Pulses circulating in tank of main store; each group represents a short number.

(*b*) Pulses in order tank; these represent an order, the address in which is 101 (five).

(*c*) Pulses in counter tank; number increases by 1 each half minor cycle.

(*d*) Coincidence waveform.

Note. The pulses which represent the least significant digits of a number come first in time and therefore appear to the left in the diagram.

the tank is specified in the order, is about to appear at the output of the tank, the five least significant digits of the number in the counter

tank are identical with the digits 0_2 to 0_6 of the order contained in the order tank. The pulses from the order tank, and also those from the counter tank, are passed into the *coincidence unit*. When suitably stimulated this unit compares the two sets of pulses arriving in each minor cycle and, when the groups corresponding to the digits concerned become identical, it proceeds to emit a waveform known as the *coincidence waveform*. This rises at the beginning of the minor cycle next after that in which the two sets of pulses were identical and remains up for half a minor cycle. Only one coincidence waveform is produced after each stimulation of the coincidence unit. The description just given may be made clearer by referring to Fig. 2·5, which shows the pulses circulating in the various tanks, or, more precisely, gives in somewhat stylized form a plot against time of the voltage output from each tank. The pulses representing the less significant digits of each number come before those representing the more significant digits, and therefore appear to the left in the diagram. The action of the coincidence unit is just the same whether the number is being transferred into or from the store, but the coincidence waveform is used to operate the input gate of the appropriate tank in one case and the output gate in the other. When a number is being transferred into the store the coincidence waveform is also applied, in the reverse form, to the clear gate, so that any number already in the storage location concerned is cleared at the same time as the new number is put in.

The digits 0_7 to 0_{11} (tank number) are, like the function digits, applied to a static register whose outputs are connected to a decoder. This decoder is also fed with an output from the order coder which indicates whether the order calls for a number to be transferred to the store or from the store. Thirty-two out of the resulting 64 outputs of the decoder are connected to the input and clear gates of the 32 tanks and the remaining 32 to the output gates. Since it is necessary to arrange that the tank input or output gates shall be open only for the required part of the major cycle, the coincidence waveform is also connected to the decoder. The appropriate decoder output rises when the coincidence waveform rises, and falls when the coincidence waveform falls. In this respect the arrangement differs from that for the function digits, where the output from the decoder rises immediately after the static register is set and remains up until it is reset. The tank-number decoder may be regarded as a 64-way switch, which routes the coincidence waveform to the appropriate tank input or output gate.

The digits 0_2 and 0_1 are also applied to a static register, whose outputs are connected to the transfer unit and determine whether or not that unit introduces the delay of a half minor cycle sometimes necessary during the transfer of a short number. The output correspond-

ing to 0_1 is also applied to the coincidence unit, and if the digit 0_1 is a 1 the coincidence waveform lasts for a whole minor cycle instead of a half minor cycle; a long number is then enabled to pass into or out of the store.

When the operation called for by the order is completed, the machine proceeds to Stage I for the next order. Stage I is very similar to Stage II, except that the operation always consists of the transfer of a number from the store to the order tank. The function static register and the associated decoder and coder are not used, the waveform which opens the gate connecting the order tank to the output bus, in preparation for the passage of the order from the store, being provided by the main control unit. The address of the order to be transferred from the store is contained in the sequence control tank, and this tank, instead of the order tank, is connected to the tank number static register and the coincidence unit, through gates operated by waveforms supplied by the main control unit. In addition to providing the various waveforms referred to, the main control unit causes the coincidence unit to be stimulated. In due course the coincidence waveform is emitted and the order passes from the store to the order tank. The static registers are then reset by a pulse from the coincidence unit. The same pulse is applied to the main control unit and initiates Stage II.

Certain orders, for example left- or right-shift, do not involve the store, and in these cases stimulation of the coincidence unit during Stage II is prevented. When the operation called for by an order is complete an *end pulse* is emitted; this comes from the main control unit, the arithmetic unit, the input unit, or the output unit, according to the type of order. In all cases the end pulse is applied to the main control unit, where it acts as a starting pulse for the next operation.

When the order which passes into the order tank during Stage I is a conditional order, and the condition is satisfied, a connexion is established during Stage II between the order tank and the sequence control tank; the order then passes into the sequence control tank and replaces whatever was there before. The result of this operation is that the sequence control tank now contains the address specified in the conditional order, so that during the next Stage I the order to be executed is taken from that address. The function digits of the conditional order pass into the sequence control tank along with the address but, since the function static register is not used during Stage I, the presence or absence of these digits is immaterial.

The Arithmetic Unit of the EDSAC

All registers in the arithmetic unit are composed of short mercury tanks. The accumulator consists of two short tanks connected in series with an adder and a shifting unit whose function will be

described shortly; the total delay round the circuit is two minor cycles, so that it is possible for all the digits which result from the multiplication of two 34-digit numbers to be stored. Any number which arrives in pulse form in the correct phase at the other input of the adder will be added to the number standing in the accumulator. The shifting unit enables the total delay in the circuit to be increased or decreased by one pulse interval and is used in connexion with right- and left-shift orders. A shift of one place to the left (that is, multiplication by 2) is effected by increasing the delay for one rotation time by an amount equal to one pulse interval. A shift to the right is similarly performed by decreasing the delay, but here a complication arises through the use of complements to represent negative numbers. If a negative number is shifted to the right it is necessary for an extra pulse to be supplied after the sign digit; for example, 1·01101 when shifted one place to the right becomes 1·101101.

A T order transfers only the 34 most significant digits (plus the sign digit) from the accumulator to the store; these are taken from one tank in odd minor cycles and from the other in even minor cycles. The less significant digits in the accumulator may be transferred to the store, if desired, by programming a preliminary left shift. If a single tank had been used for the accumulator circuit, instead of two shorter tanks in series, the 34 more significant digits would have been available only in every other minor cycle, and direct transfer to the store in the intervening minor cycles would not have been possible.

Although numbers can be taken out of the accumulator in odd or even minor cycles, they can be added into it only during odd minor cycles. An add order, therefore, causes the transfer of a number from

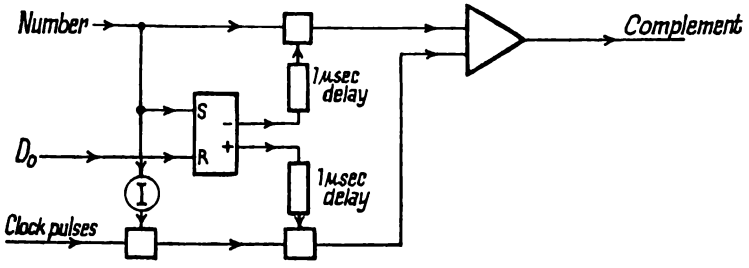


Fig. 2·6.—Logical diagram of the complementing circuit.

the store to the multiplicand tank, from which it is added automatically into the accumulator during the next odd minor cycle. Subtraction is effected in a similar way, except that the number is complemented on its way to the accumulator. This is done by means of the circuit illustrated in Fig. 2·6. It will be seen that the first pulse of a

number to arrive—this pulse represents the least significant non-zero digit—passes through unchanged, but in doing so sets the flip-flop. The negative-going output of this flip-flop is applied to a gate which closes the direct path through the unit, while the positive-going output is applied to a gate which opens an alternative path through a circuit which replaces pulses by blank spaces and blank spaces by pulses. When all the pulses of the number have passed through, the flip-flop is reset by a D_0 pulse.

Fig. 2·7 is a simplified block diagram of the arithmetic unit, de-

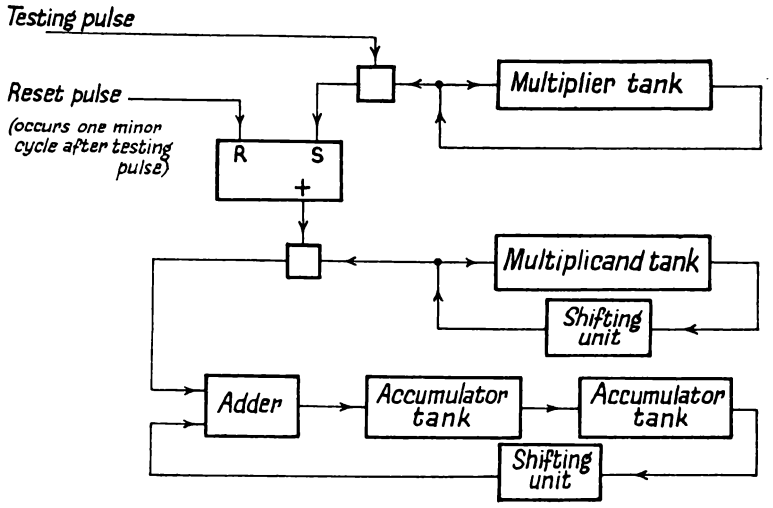


Fig. 2·7.—Simplified block diagram of the arithmetic unit of the EDSAC designed to illustrate the method of multiplication.

signed to show the way in which multiplication is performed by successive addition and shifting. At the beginning of the operation the multiplier and the multiplicand are in their respective short tanks. During the multiplication cycle, each digit of the multiplier is examined in turn, beginning with the least significant. When the digit examined is a 1, the number in the multiplicand tank is added into the accumulator; when it is a 0, no addition takes place. In each case the number in the multiplicand tank is shifted one place to the left. The method by which the multiplier digits are examined, and the addition of the multiplicand effected, is shown in Fig. 2·7. A testing pulse, which occurs at the same time in the minor cycle as the pulse representing the digit to be examined, is applied to one input of a gate, while the pulses from the multiplier tank are applied to the other input. If the digit is a 1, the testing pulse passes through the gate and

sets a flip-flop, which is then reset after one minor cycle. The output of the flip-flop controls a gate which connects the output of the multiplicand tank to the input of the adder. The result is that the number in the multiplicand tank is added once to the number in the accumulator. Shifting of the number in the multiplicand tank is effected by means of a shifting unit which introduces an extra delay, equal to one pulse interval, into the circuit when required. It should be noted that, in the EDSAC, the multiplicand only is shifted during multiplication, the multiplier and the number in the accumulator remaining as they are.

The above description is incomplete if one or both of the numbers being multiplied together are negative. When the multiplicand is negative and is being added into the accumulator in any position other than the extreme left, extra 1s are supplied as in a right-shift operation. The sign digit of the multiplier is examined in the same way as the other digits, but if it is a 1 the multiplicand is complemented before being added into the accumulator. This is equivalent to treating the sign digit as if it were -1, and it may easily be verified that the correct value for the product is obtained. The method will be recognized as a combination of two of the methods given on p. 42. In order to avoid complicating the block diagram in Fig. 2·7, the circuits required for dealing with negative numbers are not shown.

The waveforms required for controlling the process of multiplication are supplied by the *multiplier control unit*. This unit contains a short tank, one minor cycle long, called the *timing control tank*, into which a single pulse is put at the beginning of the multiplication. At every second circulation this pulse is retarded by a time equal to one pulse interval; it is used to form the testing pulse for the multiplier, together with the other pulses and gating waveforms required during the multiplication sequence. When all the digits of the multiplier have been dealt with, the pulse comes back into its original position in the minor cycle, and thereby gives an indication that the multiplication is complete and causes the emission of an end pulse.

The Input and Output Systems of the EDSAC

Standard teleprinter tape is used for input to the EDSAC. There are five positions across the tape in which holes may be punched. The presence of a hole in any position except the extreme left-hand one indicates a 1; the absence of a hole indicates a 0. In the case of the left-hand position these indications are reversed, a hole indicating a 0 and the absence of a hole a 1. Each row of holes and blanks thus represents a group of 5 (binary) digits. The left-hand position is treated differently from the others in order that the number 00000 shall be represented by a hole instead of by blank tape. The basic input operation is to transfer the digits represented by a single row of

holes on the tape to the store, where they appear as the five least significant digits of a short (17 digit) number, the other digits being 0. The input order which causes this operation to be performed is as follows:

I *n* F read the next character from the tape and place the resulting five-digit binary number, multiplied by 2^{-16}, in storage location *n*.

The punched tape is prepared by means of a teleprinter keyboard perforator. The code is not that used in telegraphy, however, but is specially chosen so that the keys labelled with the numbers 0 to 9 punch those numbers in binary form; for example, 5 is punched as 00101 and 6 as 00110. Keys labelled with letters punch the numerical values of those letters in binary form; for example, the key marked A punches the binary number corresponding to *a*. Some of the keys are labelled with numbers and letters, as on an ordinary teleprinter keyboard, and there is no difference as far as the tape is concerned between, for example, the symbols E and 3.

This system lends itself well to the input of numbers expressed in decimal notation. One row of holes on the tape is used for each decimal digit, and the first stage in the conversion of a number to the scale of two is performed by the keyboard perforator, which punches the decimal digits in their binary form. When the tape is read these digits pass into the machine as binary numbers, and to complete the conversion it is only necessary to multiply them by appropriate powers of ten and to add together the results. This calculation is carried out in the scale of two within the machine, which acts in just the same way as it does when performing any other kind of calculation.

The output system may be described as the opposite of that used for input. Printing is done by means of a teleprinter, which accepts information in the form of five-digit binary numbers and prints the corresponding figures or letters. It has a shift mechanism similar to that on an ordinary typewriter, the position of which determines whether a letter or a figure (or other symbol) is to be printed. Two of the possible five-digit numbers do not cause printing to take place, but change the shift from letters to figures, or vice versa. It is thus possible to print indicating symbols or even words along with numerical results. Other five-digit numbers, when transferred to the teleprinter, cause the paper to be advanced or a new line of printing to be started, so that any desired lay-out can be obtained. The teleprinter is not connected directly to the output of the machine, but is actuated by a paper tape punched by the machine. This arrangement has the advantage of allowing a higher average output speed to be obtained. The output order is as follows:

O *n* F transfer the five most significant digits in storage location *n* to the output punch and punch the corresponding character.

Before being printed numbers must be converted to decimal form. The principle of the method by which this is done will now be explained. Consider first a number already expressed as a decimal fraction. The successive digits, beginning with the most significant, can be isolated by multiplying repeatedly by ten and removing the integral part after each multiplication. The same method is possible if the number is expressed as a binary fraction. Successive multiplications by the binary number 1010—that is, the number ten in its binary form—are performed, and the digits to the left of the binary point removed each time. These groups of digits are the binary forms of the required decimal digits in the converted number.

When the EDSAC was first put into use the teleprinter code was the same as that of the keyboard perforator. Provided that the shift mechanism of the teleprinter was in the ' figures ' position, binary numbers less than ten were printed in their correct decimal form; for example, 00101 was printed as 5, 00110 as 6, &c. This code was chosen to facilitate the printing of decimal digits obtained by the process just described. It was, however, later replaced by a new code chosen to enable errors due to faulty teleprinter operation to be readily detectable. Such codes had already been used in connexion with other digital computers. The code is given in Table 2·1, and it will be seen that each figure is represented by a five-digit binary number which contains two 1s and three 0s. If a single error in transmission occurs the resulting number will contain either three 1s or a single 1 and will, therefore, not represent a figure. Anyone looking at the printed sheet will have no difficulty in recognizing the fact that an error has occurred. Only if two compensating errors occur—and the chance of this happening is remote—will a wrong figure be printed. The use of this code provides a high degree of security against printing errors, but it introduces a complication into the binary-decimal conversion process, since, after the decimal digits have been computed in their binary form, they must be translated into the special output code.

TABLE 2·1

0	00011
1	11000
2	00110
3	10010
4	10001
5	01010
6	10100
7	00101
8	01001
9	01100

Initial Input

From what has been said, it will be appreciated that the entire action of the machine, including the operations of input and output,

takes place under the control of a programme of orders placed in the store. Once this process has started, the action of the machine can continue indefinitely, further information, if required, being taken in from the tape in a manner provided for by the programme. In order to start the action, however, it is necessary to have some auxiliary method, which is independent of the normal control sequence, for putting orders into the machine. This is done by means of a uniselector (rotary telephone switch) on which a group of orders—referred to as the *initial input routine* or the *initial orders*—are permanently wired; when the starting button is pressed the uniselector rotates and the initial input routine is automatically placed in the store with the first order in location 0. The machine then proceeds, under the control of the initial input routine, to read the orders punched on the programme tape.

Orders are punched on the programme tape exactly as written. First there is the function letter, then the address in decimal form, and finally a code letter indicating the end of the address; this code letter has so far been taken to be either F or D. It is not necessary to punch non-significant zeros in the address; for example, A 50 F is punched, not A 0050 F.

The principles on which the initial input routine works may be best explained by taking a simplified one which is capable of reading and converting to binary form orders which are punched in the form just described, except that it is assumed that all orders are terminated by F. It will become clear, however, in Chapter 3 that a practical initial input routine must perform a number of functions over and above this minimum and that certain additional features are desirable in order to facilitate the handling of subroutines. The initial input routine used in the EDSAC is in fact quite complicated, and the description of its action will be left until later. The simplified initial input routine is as follows:

		Orders		Number of box in flow diagram
	0	T 28 F	clear accumulator	
	1	H 25 F	put 10/16 in multiplier register	1
24 →	2	T 30 F	clear storage location 30	
	3	I 28 F	read, shift, and store function letter	2
	4	A 28 F		
	5	L 12 F		
	6	T 29 F		
16 →	7	I 28 F	read and test next character	3, 4
	8	A 28 F		
	9	S 26 F		
	10	E 17 F		
	11	A 26 F	decimal-binary conversion	5
	12	R 3 F		
	13	V 30 F		
	14	L 4 F		
	15	T 30 F		
	16	F 7 F		

	Orders		Number of box in flow diagram
10→ 17	T 28 F		
18	A 29 F	build up order	6
19	A 30 F		
20	T 31 F	'transfer order'; place order in final location in store	
21	A 20 F		
22	A 27 F	increase address in 'transfer order' by 1	7
23	T 20 F		
24	F 2 F	jump back to read function letter of next order	
	Numbers		
25	10/16		
26	10.2^{-16}	constants	
27	2^{-15}		
28	—	used as 'rubbish bin' and for temporary storage	
29	—	used to store function letter	
30	—	used to build up address	

A useful convention, which has been adopted above and will be used generally, is to indicate by arrows orders to which control can be transferred by other orders in the programme. The arrows are preceded by the numbers of the storage locations from which control can come. For example, in the initial input routine given above, control can be transferred to the order in 17 by the conditional order in 10. Another convention adopted is to underline those orders which transfer control unconditionally each time they are executed.

The flow diagram given in Fig. 2·8 may help the reader to follow the way in which the above initial input routine works. The rect-

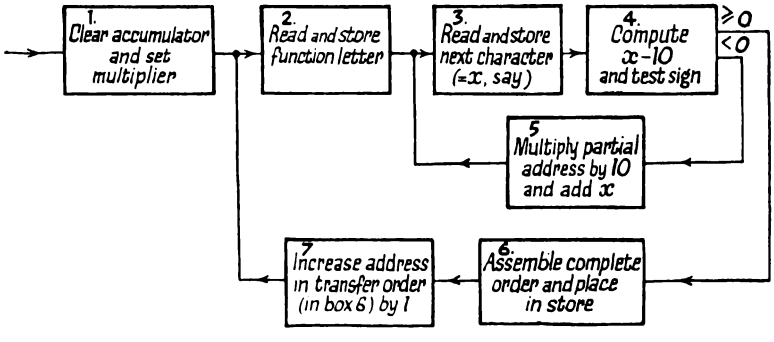

1. Clear accumulator and set multiplier
2. Read and store function letter
3. Read and store next character (=x, say)
4. Compute x−10 and test sign
≥0
<0
5 Multiply partial address by 10 and add x
7 Increase address in transfer order (in box 6) by 1
6. Assemble complete order and place in store

Fig. 2·8.—Flow diagram for a simple initial input routine.

angles, or 'boxes', with one inlet and one outlet stand for operations. Lines bearing arrows indicate the sequence in which the various operations are to be performed. The box with two outlets is used to indicate an operation involving a conditional order; if the quantity

written inside the box is positive control goes one way, and if it is negative control goes the other way.

The first character of any particular order to be read from the tape is the function letter, and this is placed in temporary storage until required for the synthesis of the complete order. As each subsequent character is read from the tape it is tested to ascertain whether it corresponds to a decimal digit or to the code letter F; this is possible because the letter F inside the machine has the form 17.2^{-16}, whereas the decimal digits have the form $d.2^{-16}$, where $d \leqslant 9$. To test a character which has just been read from the tape it is, therefore, only necessary to subtract 10.2^{-16} and to test whether the remainder is negative or non-negative. So long as successive characters, read from the tape, are found to represent decimal digits computation of the binary form of the address proceeds, but, when a code letter is detected, the complete order is assembled by adding together the address and the function digits and is placed in its final location in the store by the action of order 20. The address in this order is then increased by unity ready for the transfer of the next order. Any initial input routine must have an order corresponding to order 20, and it will be convenient on occasion to refer to it as '*the transfer order*'.

The initial input routine just given is capable of reading the programme tape and of placing the resulting orders in sequence in the store starting from storage location 31. One important provision has, however, been omitted. There is no mechanism whereby, when all the orders have been read, control may be sent to the first order of the programme, so that the calculation may be started. This omission may be rectified if a special symbol is punched after the last order on the tape. The initial input routine can then be constructed so as to perform an extra testing operation designed to detect this symbol and to send control to the start of the programme.

Special groups of symbols punched on the tape can also be used for other purposes of a similar kind, for example, to indicate the storage locations in which the orders about to be read must be placed: the initial input routine just given puts these orders automatically in location 31 onwards, and the programmer has no choice in the matter. Groups of symbols which are punched on the tape, but which do not correspond to orders or numbers intended to be placed in the store permanently as part of the programme, will be called *control combinations*.

A PARALLEL MACHINE

It would be possible to design a machine very similar to the EDSAC in logical structure, but working in the *parallel* mode. Fig. 2·2 would apply to such a machine, but the lines showing the paths of numbers to and from the store would represent groups of 35 wires, each wire

carrying one digit of a number. The main store could consist of 35 tanks each holding 512 pulses. Each tank would contain one digit of every number, and there would be no need to switch to the correct tank when passing numbers to or from the store. In order to make Fig. 2·4 apply to this machine, therefore, it would be necessary to delete the tank number static register and decoder, and to show all the address digits of an order connected to the coincidence unit. The coincidence unit would produce a single 'coincidence pulse', instead of a coincidence waveform of one minor cycle duration.

In the arithmetic unit the accumulator would consist of a static register comprising 69 flip-flops. Provision would need to be made for shifting the number in the accumulator to the right or the left, and one way of doing this would be to provide a second set of 69 flip-flops with direct and oblique connexions to the flip-flops in the accumulator register in the way shown diagrammatically in Fig. 2·9. During a shifting operation the number would be first transferred to the

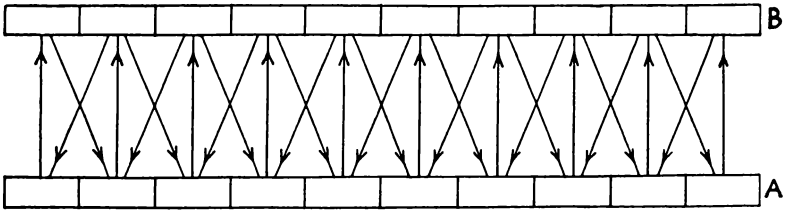


Fig. 2·9.—A parallel shifting register.

second register (*B*) and then transferred back obliquely (to the right or left as the case might be) to the accumulator register (*A*).

A register which has provision for shifting the number contained in it is known as a *shifting register*. The one just described is made up of two ordinary registers with suitable cross connexions. If desired, a connexion can be made between the most significant and least significant ends of a shifting register, so that the number can be shifted around indefinitely without being lost.

Since each digit held in a register requires a flip-flop with its associated gating circuits, there is a considerable incentive for the designer of a parallel machine to reduce the number and size of the registers as far as possible. For this reason it is better to use a method of multiplication in which the number in the accumulator is shifted after each step, instead of one in which the multiplicand is shifted. An ordinary single-length register can then be used to hold the multiplicand, instead of the double-length shifting register which would otherwise be necessary.

In a parallel machine there must be one adding circuit per binary

digit. An example of such a circuit has already been given in Fig. 1·21. Fig. 2·10 shows how the accumulator might be arranged in the hypothetical parallel machine now being described. *A* is the accumulator register proper, and *B* is the extra register necessary to convert it into a shifting register, although in order to simplify the diagram only the gates permitting direct transfer from *B* to *A* are shown. It will be observed that the leads carrying the digits of the incident number are connected directly to the adding circuits, as are the positive-going outputs of the respective flip-flops of register *A*. The operation of adding the incident number to the number standing in Register *A* is per-

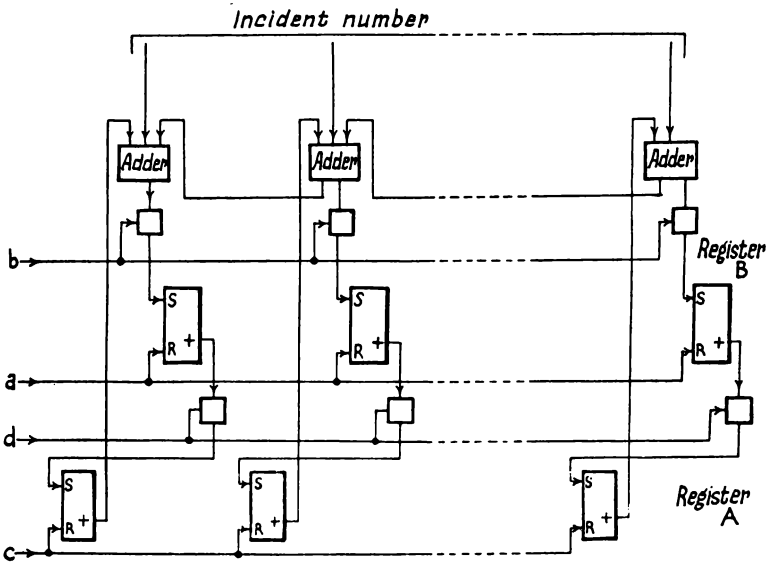


Fig. 2·10.—A parallel arithmetic unit.

formed as follows. First a pulse is applied to the lead marked *a* in order to clear Register *B*. Secondly, the gates controlling the outputs of the adding circuits are opened by means of a pulse applied to lead *b*; the sum then becomes set up on Register *B*. Thirdly, Register *A* is cleared by a pulse applied to lead *c* and, finally, the number in Register *B* is transferred to Register *A* via a set of gates which are opened by a pulse applied to lead *d*.

In order to make multiplication possible, means must be provided to sense the multiplier digits one after the other. A simple way of doing this is to make the multiplier register into a shifting register and to connect the sensing circuit to one of the end digits, say the least significant. The multiplier is then shifted one place to the right after

each step of the multiplication, so that the successive digits are brought to the sensing station in the correct order.

In the machine designed at the Institute for Advanced Study, Princeton, the provision of a separate multiplier register is avoided by using the least significant half of the accumulator to contain the multiplier at the beginning of a multiplication, the more significant half of the accumulator containing zeros. The sensing circuit for the multiplier digits is connected to the least significant flip-flop of the accumulator register. During the first stage of a multiplication, the multiplicand is added into the more significant half of the accumulator, or not added, according to whether the least significant digit of the multiplier is a 1 or a 0; the content of the accumulator (including the multiplier) is then shifted to the right. This brings the least significant digit but one of the multiplier to the sensing station and the process is repeated. It will be seen that the digits of the multiplier are gradually shifted off the end of the accumulator register as they are used, and that the product gradually fills up the accumulator.

It should be noted that the multiplier is lost when a multiplication is performed and must be reinstated if a second multiplication with the same multiplier is required. Also, if the sum of a series of products is required, one or two extra orders must be included in the programme after each multiplication in order to preserve the contents of the less significant half of the accumulator, which would otherwise be lost when the new multiplier was introduced. This is, however, not a very serious drawback to the system in view of the significant saving in the size of the arithmetic unit which it makes possible.

A logical diagram of a multiplier designed on the above principles is given in Fig. 2·11. Only gates essential for the process of multiplication are shown. Other gates would be needed in a complete arithmetic unit with full facilities for addition, subtraction, multiplication, and shifting. The multiplier shown in Fig. 2·11 contains three registers formed of flip-flops. Register *C* holds the multiplicand, which is assumed to be placed there before the multiplication begins. Register *A* is the accumulator proper, and Register *B* is the extra register provided to make shifting possible. Initially, the multiplier occupies the less significant half of Register *A*, while the more significant half contains zeros. In order to cause a multiplication to take place, a sequence of 35 suitably spaced control pulses must be applied to the input at the bottom left-hand corner of the diagram; the circuits necessary for generating these pulses, and for indicating when the multiplication is complete, are not shown in the diagram. The control pulses enter a circuit consisting of three delay units connected in series, each giving a delay longer than the duration of a pulse. Corresponding to each input pulse there are thus four pulses,

spaced in time, available for effecting the successive steps of the multiplication.

It will be observed that the gates G_1 and G_2 are connected to the least significant flip-flop of Register A in such a way that, if this flip-flop contains a 1, then G_1 is open, but, if it contains a 0, then G_2 is open. The sequence of four steps required for dealing with the first digit of the multiplier is as follows. The first control pulse passes along the lead marked a to the reset line of Register B, which is thus cleared. The pulse emerging from the first delay unit passes along lead b and, if G_1 is open, causes the sum of the numbers standing in Registers A and C to be transferred to Register B, or, if G_2 is open, causes the number in Register A to be transferred as it stands to Register B. After one or other of these actions has taken place, the pulse emerging from the second delay unit passes along lead c to the reset line of Register A and causes that register to be cleared. Finally, after a further interval, the pulse emerging from the third delay unit passes along the lead marked d and opens a set of gates through which the number standing in Register B is transferred to Register A. A similar cycle of operations is performed for each of the 35 control pulses. Even when allowance is made for the circuit required to generate the 35 pulses, it will be seen that the control circuits required for a parallel multiplier are quite simple.

When the adder is used sufficient time must be allowed for carries to be transmitted when necessary from one end of the register to the other. No similar necessity arises in the case of a direct transfer, and if suitable control circuits—more complicated than those shown in Fig. 2·11—are provided, the multiplier digits which are 0s can be dealt with more rapidly than those which are 1s.

If negative numbers are represented by complements, means analogous to those used in the EDSAC must be provided to handle the sign digit. For example, it is easy to arrange that during right shift the flip-flop on the extreme left of the register does not change its state. This causes extra 1s to be introduced in the correct manner when the number in the register is negative.

REMARKS ON THE GENERAL PRINCIPLES OF MACHINE DESIGN

The reader who has followed the functional description of the EDSAC, and the remarks on parallel machines which followed it, will have a good idea of the general lay-out of any universal digital computing machine. The component parts of any machine were given on page 6 as being store, arithmetic unit, input, output, and control. The division between these components is to some extent arbitrary, particularly that between the control and the other units. Usage varies a good deal from one design group to another; in this

book the control will be taken to include everything which is not strictly concerned with the handling of numbers, the only exception being that the circuits which enable access to be had to a given storage location will be regarded as forming part of the store.

The most obvious division of computing machines is between those which are serial in operation and those which are parallel. Some general remarks may be made by way of contrasting serial and parallel machines, but the subject is an endless one in that many different variants of each kind of machine are possible. I do not consider that the question of whether it is better to build a serial or a parallel machine will ever be finally answered; each type has its advantages, and the final decision will always be dependent on the designer's personal preferences, and to some extent on his terms of reference. Also, as time goes on, the balance of advantage and disadvantage will swing as improvements are made in this or that component, or as entirely new components become available. Certain fundamental comparisons between serial and parallel machines can, however, be drawn readily enough.

The arithmetic unit of a parallel machine tends to be much larger than the arithmetic unit of a serial machine, since a separate flip-flop or equivalent circuit element must be provided to accommodate each digit in the various registers. Moreover, since addition of the digits takes place simultaneously, or nearly so, a separate adder must be provided for each digit. The disparity between the sizes of the arithmetic units in serial and parallel machines increases as the fundamental number length is increased; for example, in the EDSAC the number length could be doubled as far as the arithmetic unit is concerned simply by replacing the existing short mercury tanks by other tanks of twice the length, whereas a parallel arithmetic unit would be exactly doubled in size by such a change. If a fundamental number length of 10 or 12 binary digits were acceptable to users, then perhaps the size of the arithmetic unit would not be greatly affected by the choice between serial and parallel modes of operation.

The control, on the other hand, is much simpler in a parallel machine. In the first place the necessity for the generation of a set of digit pulses does not arise, and secondly, the timing is much simpler, since a number can be moved from one part of the machine to another, or an addition performed, by the application of a single pulse to a set of gates, whereas in a serial machine a gating waveform of some definite duration is required. These simpler requirements lead, in the case of a parallel machine, to simpler control circuits.

Although the division of machines into those which work in the serial mode and those which work in the parallel mode is perhaps the most obvious way of classifying them as regards logical design, from a technical point of view another mode of classification is of equal im-

portance. This is whether the machine is synchronous or asynchronous in operation.

The EDSAC is an example of a synchronous machine. It is controlled by a clock pulse oscillator from which other pulses defining minor and major cycles are derived. All switching waveforms rise and fall at times defined in relation to these pulses. Consider, for example, the waveform required to add the content of the multiplicand tank to the accumulator; this waveform must rise in synchronism with a minor cycle pulse, and it must stay up for a complete minor cycle. Both rise and fall must not take longer than a specified time.

In an asynchronous machine a wide tolerance is allowable on both the shape and duration of control pulses, and the time at which a particular pulse occurs is immaterial, provided that it does not rise until the preceding operation is finished. The designer does not, therefore, need to take steps to ensure that the control waveforms are accurately phased in relation to the clock pulses. Such a machine could be so designed that a pulse emitted on the completion of one operation would cause the next operation to begin. Usually, however, even when a machine is designed to be asynchronous in the sense defined here, it is, as a matter of convenience, driven from a clock pulse oscillator to which it is, strictly speaking, synchronized. In these circumstances the use of the term 'asynchronous' is a little misleading.

A parallel arithmetic unit contains a good deal more equipment—that is, a larger number of circuit elements such as flip-flops and gates—than does a serial arithmetic unit, but its overall speed is greater. The comparison here is assumed to be made between parallel and serial arithmetic units built up from logical elements having similar properties as regards speed of operation. It is often the case that a higher overall speed of operation can be obtained by increasing the amount of equipment in a machine and, conversely, a machine can often be simplified and reduced in size if a lower speed of operation is accepted. In the parallel arithmetic unit the extra speed is obtained by performing simultaneously in separate channels operations which would otherwise be performed in time sequence in a single channel. A less obvious example is provided by the use, in machines such as the EDSAC, of a subroutine to perform the operation of division instead of a built-in divider. This means that division takes ten or more times as long as it otherwise would, but, since division can be made a comparatively infrequent operation, the overall loss of operating speed is not high, and can sometimes be accepted as a fair price to pay for the elimination of the control circuits necessary to perform division.

In the development of automatic digital computers which took place in the 10 years following the completion of the Automatic

Sequence Controlled Calculator, there was a general tendency to rationalize and simplify the design. This came about partly as a result of technical developments, particularly in high-speed storage, and partly as a result of theoretical advances in logical design. An example of rationalization is the elimination, wherever possible, of specialized units and the handing over of their functions to the main units of the machine. Early machines, such as the Automatic Sequence Controlled Calculator and the large relay machines, were provided with special devices whereby tables of functions could be put in the machine and automatically consulted and, if necessary, interpolated. Frequently special units were provided for calculating such functions as sines, logarithms, &c. In modern electronic machines, on the other hand, it is assumed that values of special functions will be calculated whenever they are required by means of power series or by some similar method, and that when, exceptionally, a table of a function is required, it will be placed in the store along with other numerical information and interpolated by means of a subroutine.

The tendency to simplification was accentuated by the willingness of designers to use some of the gain in speed, which the development of fast basic circuits made possible, to simplify the machine, instead of using it all to obtain a higher operating speed. Now that the possibilities of obtaining higher operating speeds by the use of intrinsically faster circuits, or by improved logical design, are becoming exhausted there are signs of a reversal of this trend, and it is to be expected that in the future designers will be more ready to include extra equipment in their machines, if they can obtain thereby a significant increase in speed. Technical advances which improve the reliability of large machines will obviously favour this development, as also will the development of smaller and more efficient logical elements. If, however, the time ever comes when a machine 10,000 times as fast as existing machines is being designed, then probably the designer will wonder whether he should even provide a multiplier or whether he should be content with an adder and leave it to the programmer to perform multiplication with the aid of a subroutine. In a machine in which the micro-microsecond rather than the microsecond were the natural unit of time—if such a thing can be imagined—the arithmetic unit might very well have provision only for the logical operations of 'and' and 'or'.

Methods of programme organization designed to facilitate the handling of subroutines will be discussed in the next chapter, but it may be pointed out here that the decision whether to provide a piece of equipment to perform a particular function, or whether to make use of a subroutine, is one which is to be taken largely on engineering grounds. It does not make very much difference to the programmer whether he has to write down the orders necessary to call some special

unit of the machine into action or the orders necessary to call in a subroutine for performing the same function. In fact, if the subroutine is held permanently in a part of the auxiliary store reserved for the purpose, the programmer need hardly know that he is using a subroutine rather than a built-in unit.

It must be remembered that the replacement of a built-in unit by a subroutine is not an entire gain, from the point of view of the overall size of the machine, since additional storage capacity must be provided to hold the subroutine. In machines which have large stores already this is not a point of great importance, but where storage is expensive (in a relay machine for example) it may well swing the balance in favour of the built-in unit.

In the following sections various special topics connected with the logical design of computers will be discussed.

Order Interpretation

This section deals with the way in which the function digits of an order are used to produce switching waveforms for setting up the correct conditions in the various parts of the machine. The system used in the EDSAC was described on p. 50. It will be remembered that the five function digits are set up on a static register which is permanently connected to a decoding tree, and that one output of the decoding tree rises in potential for each type of order. Since, however, certain gates in the machine have to be opened for more than one type of order it is necessary to follow the decoding tree with a coder. This is so constructed that, to take an example, the output which controls the gate leading to the store rises in potential for all orders which require this gate to be open.

The system just referred to enables the most economical use to be made of a limited number of function digits; for example, with five function digits up to 32 distinct orders can be provided. There are, moreover, no restrictions on the choice of combinations of function digits to represent the various orders, so that the adoption of a mnemonic code is facilitated. On the other hand, the coding and decoding system of a machine can be greatly simplified (at the expense of using more function digits) by associating particular function digits with particular classes of orders. For example, the presence or absence of one of the function digits can indicate whether the arithmetic unit is to be set up to perform an addition or a subtraction, another can indicate whether the order calls for the transfer of a number to or from the store, and a third can be used to indicate whether the accumulator is to be cleared or not cleared at the end of the operation. In all cases, the flip-flops on which these digits become set up are connected directly to the appropriate gates in the machine. In terms of Fig. 2·4, this may be expressed by saying that some of the

connexions shown coming from the coder would come direct from the function static register, by-passing the decoder and coder altogether. This system has been carried almost to the logical extreme in the machine built at the Institute for Advanced Study in which there are 10 function digits. A characteristic of machines built on this principle is that they have a large number of available orders, corresponding to the large number of function digits used. Not all combinations of function digits, however, correspond to useful orders.

The system has the disadvantage that more function digits are used than are strictly necessary to convey the required amount of information. This means that more storage capacity, measured in binary digits, is required to hold a given programme of orders. However, in the machine mentioned above, the basic word-length was chosen with the representation of numbers in mind, and plenty of digits were available for orders. The result was that all digits in each order were made use of instead of some being left spare. Whether the scheme would still be attractive if the conflict between the length of word most suited to the storage of numbers and that most suited to the storage of orders were resolved in some other manner, so that spare digits were not available, is a matter for discussion. It might be argued that the disadvantage of having to provide more storage capacity to hold a given number of orders would be more than offset by the elimination, or partial elimination, of the decoding and coding units.

Apart from the considerations connected with the decoding of orders that have just been explained, there are few matters of general principle connected with the logical design of the control circuits of an electronic computing machine which call for discussion. The control sequence of any machine using a single-address code necessarily proceeds in two stages, the first being the extraction of an order from the store, and the second the execution of that order. It is necessary to have a sequence control counter, the number in which determines the location from which the next order is to come. Every time an order is executed the number in this counter is increased by 1, except when the order causes a transfer of control. The number in the sequence control counter is then modified in some other way; usually it is replaced by the number specified in the order. The control sequence of a machine using a two- or three-address order code proceeds in two or more stages, depending on whether the addresses are dealt with concurrently or sequentially. If the order code is of the variety in which each order contains a statement of the address of the next order a sequence control counter is not necessary; instead a simpler register suffices. This receives the address of the next order each time an order is executed; if the order happens to be a conditional one the register receives either of two addresses according to whether the condition is satisfied or not.

The Modification of Orders

The fact that in EDVAC-type machines it is possible to modify orders by adding numbers to them, makes it possible, as was shown above, for the same sequence of orders to be used over and over again but with a slightly different action each time. This was not possible in earlier machines, in which orders punched on endless loops of tape could be used as many times as desired but always with identically the same action. There is, however, another method by which the same facility may be obtained without it being necessary to modify the orders themselves. This is by providing orders with what will here be called 'algebraic' addresses. A number of registers—which will be referred to as α, β, γ, &c.—are provided in which numbers can be placed by the programme. An example of an order with an algebraic address, written in a manner suggested by the EDSAC code, is then:

A (α) F Add the number in storage location n into the accumulator, where n is the number contained in register α.

The action of a sequence of orders in which orders of this kind are included can be modified by changing the numbers in registers α, β, γ, &c. This system of programming may be illustrated by the example already used on p. 46, namely, the addition of a sequence of 100 numbers placed in the store from storage location 50 onwards and the placing of the sum in storage location 200. The arithmetic orders (corresponding to orders 12, 13, and 14 of the sequence given on p. 47) are as follows:

A 200 F
A (α) F
T 200 F

The number in register α is originally set to 50, and is increased by 1 each time the sequence of orders is repeated. After each repetition the number in register α is tested, and when it is found to have reached 149 the cycle of repetitions is brought to an end.

The philosophy lying behind the design of a machine using algebraic addresses is quite different from that lying behind a machine of the EDVAC type, and it is in this connexion that the greatest differences of opinion between machine designers are to be found. If algebraic addresses are used, orders and numbers are kept quite separate, and it is most natural that they should be accommodated in entirely separate stores. The order store must be fairly large, since the programmer has little opportunity to use programme-shortening techniques, and programmes are likely to be long compared with those used in EDVAC-type machines. Orders with algebraic addresses are used in the two latest machines (Mark III and Mark IV) built at the Computation Laboratory at Harvard.

If the orders are invariant, as they are in a machine in which algebraic addresses are used, it does not matter if it takes some time for the programme to be read into the order store, provided that the orders can be read back at high speed. It might be thought that this would allow a cheaper and less complicated form of store to be used for the orders. At the present time, however, the only forms of store available for use in a really high-speed machine can all be written into as quickly as they can be read from. This particular advantage of using algebraic addresses is, therefore, significant only in the case of relay, or other comparatively slow, machines, when it enables punched tape to be used for storage of orders. However, the situation might be greatly changed if some other kind of store with different properties were invented.

B-Registers

An important device, which was originated at Manchester University, for facilitating the modification of orders will now be described. In an ordinary programme the modifications made to orders consist almost entirely of adding numbers to them. In the system being described a number of special registers, known as *B-lines* or *B-registers*, are provided, and the content of one of these registers is added to every order before it is executed. The programmer specifies the B-register whose content is to be added by assigning appropriate values to a group of binary digits which are set aside in every order for this purpose. One of the B-registers always contains 0, so that the effect of adding the content of this B-register is to leave the order unmodified. Orders which have the content of a B-register (other than the one containing zero) added to them are here called 'B-modified' orders. The order code of the machine contains orders by which the programmer may transfer numbers from the store to any of the B-registers, and other orders by means of which he may subtract a number, taken from the store, from the number standing in any B-register. (It would be possible also to provide an order which would enable a number, taken from the store, to be added to one contained in a B-register.) There is also a conditional order which transfers control if the number standing in a specified B-register is positive or zero.

The example given above would be programmed for a machine fitted with B-registers in the following manner (b_1 stands for the content of the first B-register and each line corresponds to one order):

Set $b_1 = 99$
Add C($50 + b_1$) into the accumulator (this is a B-modified order)
Subtract 1 from b_1
Test b_1; if $\geqslant 0$ return to second order, otherwise proceed
Transfer content of the accumulator to 200.

It will be seen that the use of B-registers in conjunction with the special orders described above enables the modification of orders and counting to be performed without disturbing the content of the accumulator. Furthermore, a whole group of orders in a repetitive loop can have their addresses increased by a given amount merely by increasing the number in one B-register, so that the necessity for individual modification of the orders is avoided. For example, suppose it is required to transfer the content of storage location 99 to storage location 100, the content of 98 to 99, &c., the last transfer being from 50 to 51. If, initially, b_1 is set equal to 49, a repetitive loop for this purpose would consist of the two B-modified orders,

Add C($50 + b_1$) into accumulator
Transfer C(Acc) to $51 + b_1$,

followed by orders for decreasing b_1 and testing for the end of the cycle. A further advantage of the use of B-registers is that, since the orders in the programme itself remain unmodified, all that has to be done to reset a sequence of orders which are required for use a second time is to place the new starting numbers in the B-registers.

Not only does the provision of B-registers help to simplify the task of the programmer, but it also decreases the number of orders that are necessary in a given programme and reduces the amount of machine time required to run it. Part of the gain in speed results from the simplifying of the programme and part from the fact that the access time of the B-registers is short compared with that of locations in the main store. If a storage system with very rapid access were used this latter advantage would naturally be less marked.

In the machines built at Manchester orders and numbers are contained in the same store and the programmer is free, as with other machines of the EDVAC-type, to modify orders in the arithmetic unit and to plant orders in the programme if he wishes. The provision of the B-registers does not, therefore, enable anything to be done which could not be done otherwise, but it does enable faster operation to be obtained.

DECIMAL MACHINES

It is desirable to say something at this stage about decimal machines constructed with the aid of binary elements, although a full discussion of this subject will be postponed until Chapter 6. In the ENIAC, which was the first machine of this kind to be built, each decimal digit is conveyed from one part of the machine to another along a single wire, which carries a sequence of pulses equal in number to the value of the digit being transmitted. Addition is performed by means of electronic ring counters.

An alternative system would be to transmit each decimal digit

along a group of 10 wires, one, and only one, of which carried a (single) pulse when a digit was being transmitted. An adder for use in a machine working on this principle could be constructed along similar lines to the binary adder described on pp. 34–5, although the decimal addition table built into it would naturally be more complicated than the binary addition table. An adder working in this system, and designed for use in a machine in which electro-magnetic relays are used for storage and switching elements, is shown in Fig. 2·12. Each wire

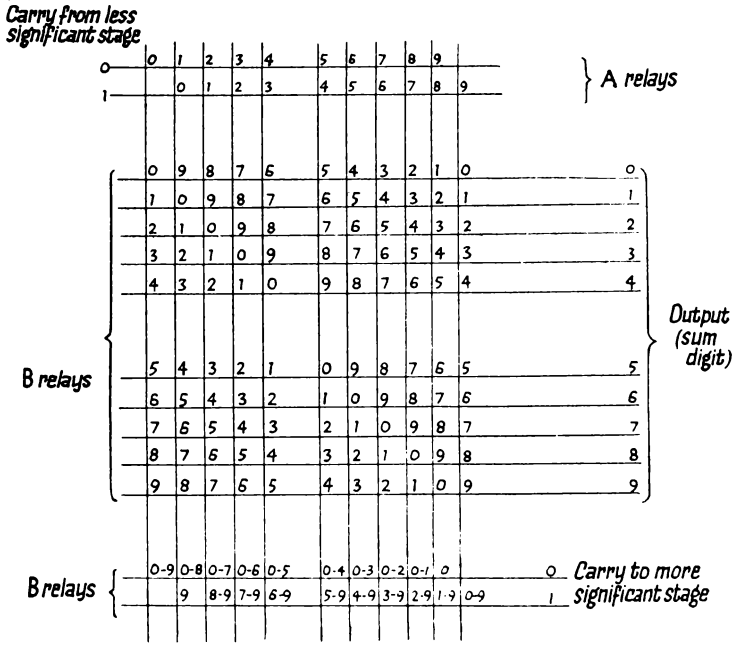


Fig. 2·12.—A relay adder (scale of ten).

of the two groups carrying the incoming digits is connected to the coil of a relay; the other end of each coil is earthed. When a digit is transmitted one lead in each group rises in potential and the corresponding relays close. In Fig. 2·12 the two groups of relays are designated by the letters *A* and *B*; if, for example, the incoming digits are 5 and 3 respectively, relay 5 of the *A* group and relay 3 of the *B* group close. To show the connexions to the relay contacts in the usual way would make the diagram difficult to follow; instead, leads which are connected together when a relay closes are shown crossing one another on the diagram with the number of the relay concerned written near the intersection. It is easy to verify that the arrangement

shown in the diagram gives the correct values for the sum and carry digits in all cases.

The system of representation of decimal digits just described (which will be referred to as the one-out-of-ten system) is equivalent to representing the successive decimal digits by the binary numbers 0000000001, 0000000010, 0000000100, &c. If each wire is capable of carrying a 0 or a 1 irrespectively of what the other wires are carrying, it is, however, clearly wasteful to use 10 wires to carry a single decimal digit, since only 10 of the 2^{10}, or 1,024, possible combinations of digits (those with nine 0s and a single 1) are used. The smallest number of binary digits which can be used to transmit a decimal digit is four, and even here there is some redundancy, since 16 combinations are possible and only 10 are used.

In a machine in which four binary digits are used to represent a decimal digit, the designer is free to choose any 10 of the 16 possible combinations to represent decimal digits. The most obvious course to adopt is to represent each decimal digit by the corresponding binary number. This system is shown in the second column of Table 2·2, and it has the advantage that operations based on counting procedures—for example, addition, or conversion to or from the scale of 10—can be performed simply, because the binary groups occur in their natural order. A variant, proposed by Stibitz, is known as the *excess-three* code. This is shown in the third column of the table, and it will be seen that each decimal digit is represented by the corresponding binary number with three units added. This system has similar advantages to the one just described, and in addition it possesses the property that any digit can be converted into its nines complement by replacing the 1s by 0s and the 0s by 1s.

TABLE 2·2

0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

If a four-digit binary code is used, $4n$ binary digits are needed to represent a number with n decimal digits. These may be transmitted from one part of the machine to another in various ways. In a fully serial machine the $4n$ digits would be transmitted in time sequence along a single wire, while in a fully parallel machine they would be transmitted simultaneously along a group of $4n$ wires. An intermediate arrangement would be to provide n wires and to transmit the

four binary digits representing each decimal digit in time sequence along one wire; a further alternative would be to provide four wires and to transmit the four binary digits representing each decimal digit along the four wires simultaneously, the various decimal digits being transmitted in time sequence.

In addition to the binary codes given in Table 2·2, a number of others have been used, or proposed for use, in computing machines. The choice of code has a considerable influence on the design of a decimal machine, and the subject will be discussed more fully in Chapter 6.

INPUT, OUTPUT, AND AUXILIARY STORAGE

In most of the early machines to be constructed either punched paper tape or punched cards were used for input. For example, the ENIAC and the Pilot Model of the ACE used punched cards, and the EDSAC, the SEAC, the SWAC, Whirlwind I, the machines constructed at Manchester University, and the machines based on the design of the Institute for Advanced Study, Princeton, originally used paper tape. The first two machines similarly used punched cards for output, and the others used either punched tape or a directly connected printer. Magnetic tape has also been used, for example, in the UNIVAC and in the Harvard Mark III and Mark IV machines. Some notes on the various media and the speeds attainable with them are given below.

Sometimes results taken out of a digital computer may not be required as answers to a calculation but may be intended for subsequent re-input. This is frequently necessary if the machine has no auxiliary store, or if the auxiliary store is not large enough to be used for day-to-day storage of information. It is also convenient when it is expected that some considerable time will elapse before the information is required again, or when some sorting is necessary and can be better done on auxiliary equipment than in the machine itself. Again it may be desired to have the information in a form in which portions can be printed if required; otherwise temporary output, like auxiliary storage, may take place in some form not suitable for direct printing; for example, numbers may be in the binary scale. It should be noted that, while auxiliary storage can always be used instead of temporary output and re-input, the converse is not true, since it may not be known in advance when and in what order the various sections of information will again be required in the high-speed store.

In some machines the same mechanical devices are used for input and output (including temporary output) and auxiliary storage, and the distinction between the two functions becomes somewhat blurred. In the UNIVAC, for example, magnetic tape is used both for auxiliary storage and as a means of input and output, equipment being pro-

vided for printing information from tape and for writing information on the tape from a keyboard. This was comparatively simple to arrange, since this machine works in the decimal system. Information intended to be printed is, however, written on the tape with a lower packing density than information which is intended to be read back directly into the high-speed store, so that the two roles are to this extent distinguishable.

In view of the close similarity of input and output and auxiliary storage, both from the point of view of the user of the machine and from that of the engineer, it is not surprising that confusion has sometimes arisen concerning them. For example, prospective users of a computer have sometimes insisted on the necessity for a very high speed of output when what they really wanted was an efficient auxiliary store. In the same way comparisons between different machines can be misleading if the complementary nature of the roles played by input and output and auxiliary storage is not borne in mind.

Punched Paper Tape

Commercially available tape perforators and printers can be used to form the basis of an input and output system for a computer. Both five- and six-hole punched tape is used in telegraphy and seven-hole tape is used in connexion with the Flexowriter tape-controlled typewriter, although in this equipment one of the holes is used for control purposes only. Commercial telegraph tape readers may readily be connected to form the input channel of a digital computer, but the standard speed of reading used in telegraphy—about seven rows of holes per second—is too slow to be acceptable, and it is more usual to make use of specially designed photo-electric tape readers. One such tape reader is made by Ferranti Ltd., and is capable of reading tape at speeds of up to 200 rows of holes per second. When the tape reader is at rest one row of holes stands under the reading head, and the information contained in that row is instantaneously available to the computer. While the tape reader is advancing the tape, the machine can continue to compute, and only if the information contained in the next row is required before that row has reached the reading head does the computer need to wait for the tape reader.

It will be seen that a photo-electric tape reader is capable of transferring information to a computer at the rate of about 1,000–1,400 binary digits per second, depending on the design of the tape reader and on whether five-, six-, or seven-hole tape is used. If data are to be presented to the computer in decimal form it is natural to take one row of holes to represent a decimal digit; the speed attained is then of the order of 200 decimal digits per second. The tape reader just mentioned is so designed that, after reading a row of holes and advancing the tape, it stops instantly with the next row in position to be read,

unless the computer is waiting for the information contained in that row or calls for it immediately. The tape reader will do this even if the tape is moving at full speed after a number of rows have been read in quick succession. This feature makes possible very convenient and efficient programming techniques, and renders it unnecessary for the data to be divided beforehand into blocks so that spaces for stopping can be left when the tape is punched. If instantaneous stopping is not insisted on, however, higher speeds of reading can be obtained. There does not seem to be any reason why speeds of 1,000 or 1,500 rows of holes per second should not be attained; these figures correspond to 5,000–7,000 and 7,500–10,500 binary digits per second respectively.

Standard telegraph perforators may be adapted for use in the output channel of a computer, and speeds of up to 20 rows of holes per second may be readily attained. This corresponds to 100–140 binary digits per second or 20 decimal digits per second. A high-speed perforator designed by the Teletype Corporation is capable of punching paper tape at the rate of 60 rows of holes per second. This perforator is a motor-driven device, and, in order to obtain optimum performance, it is necessary to provide buffer storage between it and the computer.

Punched Cards

Standard IBM or Hollerith punched cards have eighty columns in which holes may be punched in one or more of twelve positions. Ordinarily, when decimal numbers are punched on a card, one hole only is punched in each column, and 80 decimal digits in all can be accommodated; by making full use of all the positions on the cards in which holes may be punched, as many as 960 binary digits can be accommodated. In digital computers which use punched cards for input, however, it is not usually possible to transfer information from all the 80 columns into the machine during a single passage of the card. For example, in the ACE (pilot model) it is possible to transfer information from 32 columns only, and in the more recent IBM 701 machine it is possible to transfer information from 72 columns; in both cases the columns from which information is to be taken are determined in advance by setting up a plug board. Apart from these limitations, however, natural subdivisions occurring in the data may make it inconvenient for the programmer to utilize fully the accommodation available on a card; for example, in a certain harmonic analysis of barometric observations it was convenient to punch a day's observations on a single card, although these did not quite fill the card. These considerations must be borne in mind when comparing the rates of transfer which are possible with cards with the corresponding rates which are possible with other input media.

In card readers, such as are normally used in the input channel of a digital computer, the cards are read row by row by means of a set of brushes, one brush per column. The process of reading cards is, therefore, very similar to that of reading tape, except that the number of holes or blank spaces in each row is much greater. When a row of holes has been read, and the corresponding binary number transferred to the store, it is possible for independent computation to proceed within the machine until the next row of holes is in position ready to be read. Cards differ from tape, however, in that a card cannot be halted between the reading of successive rows; once the passage of a card has been started, by means of an appropriate order included in the programme, the successive holes become available for reading at regular intervals of time, and the information contained in a given row must be either accepted or passed over.

In the IBM 701 computer cards are read at the rate of 150 per minute ($2\frac{1}{2}$ per second). If the cards are used to their best advantage information may be transferred to the high-speed store at the rate of 2,160 binary digits per second; decimal digits punched in the standard manner may be transferred at the rate of 180 per second. Nearly three-quarters of the time occupied by the passage of a card may be utilized for independent calculation.

The card punch used for output in the IBM 701 machine punches holes row by row in any 72 columns predetermined by means of a plug board. Cards are fed at the rate of 100 per minute ($1\frac{2}{3}$ per second). Binary digits can thus be punched at a rate of 1,440 per second and decimal digits at a rate of 120 per second.

In some circumstances the programmer may find it convenient to form in the store an image of the configuration of binary digits to be punched on the card, before the card is set in motion. In other circumstances, it may be more convenient for each new row of digits to be computed immediately after the previous row has been punched. In the IBM 701 machine rather more than two-thirds of the time taken for the passage of a card is available for independent computation.

Magnetic Tape

Magnetic tape may be used both for auxiliary storage and for input and output; I am here concerned with its use in the latter roles. The technical aspects of recording on magnetic tape will be discussed in Chapter 5.

It is possible to run magnetic tape at speeds of 100 in. per second or more, and to use packing densities of 100–200 binary digits per inch. Thus, a rate of transfer of 10,000–20,000 binary digits per second may be achieved from a single track; five or six tracks may be accommodated on tape $\frac{1}{2}$ in. wide. It is usually necessary, however,

to use one track for synchronizing purposes, and the presence of flaws in the coating of the tape makes it desirable for there to be some redundancy in the recorded information.

Unfortunately, if densities as high as those mentioned above are used on an output tape, quite elaborate apparatus is necessary for printing information which has been recorded by the machine. The difficulty is that, if the tape is run slowly enough for information to be read from it and printed directly by a teleprinter, or by an automatic typewriter (not more than 40 binary digits per second at the most), there is difficulty in obtaining a sufficiently strong signal from the reading head for satisfactory operation. It is therefore necessary to run the tape more rapidly and to transfer information from it into a buffer store, halting the tape periodically while the information in the buffer store is printed. If a much lower density of recording on the tape is accepted—say about 20 binary digits per inch—it is possible to run the tape at such a speed that the information from it can be printed directly. For this reason a recording density of 20 binary digits per inch (on each track) was adopted for output purposes in the UNIVAC; a similar density was used in the Harvard Mark IV machine.

The use of magnetic tape as the input and output medium for a digital computer enables large quantities of information to be transferred rapidly through a small number of channels, and is economical from the point of view of the amount of equipment needed in the computer itself. Difficulties in handling arise on account of the fact that information recorded on magnetic tape is invisible, but in spite of this it appears likely that magnetic tape will, as time goes on, find wider application for input and output purposes in digital computers.

Printers

A teleprinter or automatic typewriter directly coupled to a digital computer is too slow for satisfactory use as the principal output device, although one is sometimes provided for printing supervisory information. Directly coupled printers of the line-by-line type are useful in some applications, in which it is known that the information will not have to be put back into the machine, and in which an adequate number of copies can be obtained by the use of carbon paper. The IBM 701 computer, for example, is equipped with a printer which will print 150 lines per minute, each line containing up to 72 decimal digits or other characters. The maximum printing speed is thus 180 decimal digits per second.

Printers are also required for printing information which has been taken out of a machine on punched paper tape, punched cards, or magnetic tape. If the amount of material to be printed is relatively small, as it often is in the case of a scientific calculation, the use of

teleprinters or automatic typewriters is economic and satisfactory, but if a large amount of material is to be printed the number of parallel channels required becomes so large that a high-speed printer is cheaper, not only in first cost, but also in the cost of maintenance. For example, 26 teleprinters would be necessary to keep up with the line-by-line printer just mentioned. Much development work on really high-speed printers is in progress in various laboratories.

Photographic Recording

Information can be recorded very rapidly and with high density on photographic film, and similarly can be read back at high speed. Unfortunately the need to process film has limited severely its application as an input and output medium for digital computers. The situation is unlikely to change in the future unless marked advances are made in processing techniques.

Various printing devices have been developed in which figures and other characters are traced on the screen of a cathode-ray tube by a suitable scanning technique and photographed. These devices are very rapid in action and, if a sufficiently high typographical standard can be achieved, should find a number of applications.

Cathode-ray Display Devices

For some purposes, it is convenient to have a visual indication of the progress of a calculation in the form of a curve or other diagram, and a display device based on the use of a cathode-ray tube has been developed at M.I.T. to provide this facility. If desired, a permanent record can be obtained by photographing the display. The device contains circuits by means of which the spot on the screen of the cathode-ray tube—which is of the type which gives a long afterglow—can be positioned according to co-ordinates specified in digital form. The spot is normally blacked out, but is momentarily brightened each time it is positioned. By including suitable orders in his programme, therefore, the programmer can cause to be plotted out point by point any curve or other diagram he chooses. It is even possible to trace out numerals, and this facility is useful, both for adding a few numbers to a diagram and as a fast means of output for small quantities of numerical information.

CHECKING OF NUMERICAL RESULTS

Mechanical or electrical faults are always likely to occur in an automatic calculating machine. Some of these—for example, the failure of a valve or other component in a vital part of the control—put the machine completely out of action, or affect its behaviour so radically that it is obvious that a fault has occurred. These faults, which may be called major or catastrophic, are usually easy to find, and their

occurrence is not likely to lead to any difficulty on the part of the user, except for the loss of time that they cause. In the case of other faults, however, the machine appears to function quite normally, and the only effect of the fault is that occasional arithmetic errors are made. Faults of this kind can be due to the failure of a component in an infrequently used part of the machine, or—more often—to some slight maladjustment of some circuit in the machine which produces an error only under unfavourable circumstances. A bad connexion, such as might be caused by a dry joint or by a faulty plug or socket, can similarly produce occasional errors. Unless some method of checking is employed there is danger that these errors will go undetected. Even when no fault is present in the machine, errors may arise as a result of external circumstances, for example, the occurrence near by of a flash of lightning. Naturally the designer will do all he can to isolate the machine from interference of this kind, but it is necessary to remember that any machine, however well designed and maintained, is likely to make occasional errors.

There are two contrary points of view which may be taken by the designer of an automatic calculating machine in relation to the control of sporadic errors. On the one hand he may include in the machine itself devices which will give an alarm if some action called for by the programme fails to take place correctly. The most obvious way in which this can be done is to build part of the machine in duplicate; for example, a machine might have two multipliers with circuits for verifying that the products produced are the same. An example of a less obvious method of giving protection against error is the logical check described on p. 58 in connexion with the teleprinter code used for output from the EDSAC. On the other hand, the designer can elect to provide no automatic checking devices, but to leave it to the programmer to include in his programme such checks as he considers necessary.

At first sight it might seem that the former of these two policies is without any question the one to adopt. There are, however, disadvantages in introducing extra components into an automatic calculating machine, which is large and complicated enough without being made more so. It must be remembered that, if any part of the machine is, say, doubled in size in order to provide checking, then twice as many faults must be expected to occur in it. Thus, although the checking device may prevent an error going undetected, its inclusion will decrease the overall reliability of the machine.

Any protection against sporadic errors which can be provided by a piece of equipment can also be provided by a sequence of orders in the programme. For example, the action of the multiplier may be checked by performing each multiplication twice, with the multiplier and multiplicand interchanged the second time, and verifying that the

two results obtained are the same. Alternatively, the multiplier and multiplicand can be multiplied by k and $1/k$, respectively, where k is a constant, before the second multiplication is performed. Detailed checks of this kind, however, will be required only if the machine is decidedly unreliable. Normally checks which cover a whole section of the calculation will be sufficient. Such checks, which may be compared to the checks an accountant performs when he balances his books, are frequently available in mathematical calculations, although in other cases some form of checking by repetition must be used.

However many checking devices are built into the machine, the programmer must provide some checks himself, if only to make sure that his programme is correct; errors due to mistakes in presenting the problem to the machine clearly cannot be detected by devices for ensuring that the machine is functioning correctly. In any case, the onus of establishing that results are correct rests fairly and squarely on the shoulders of the programmer; no mathematician or theoretical physicist should be prepared to believe that a set of numerical results are correct because they are supposed to have been produced by an infallible machine. Some sort of cross check is necessary to give confidence in them. But at the same time it should be realized that no checking system, mechanical or mathematical, is infallible, since there is always the danger that compensating errors will occur. While this danger may be reduced to small proportions, it can never be wholly eliminated.

The overall checks which are available to a programmer for ensuring the correctness of his results naturally depend to a large extent on the particular problem in hand. Certain methods are, however, of wide applicability; for example, in the case of tabulation the method of differences may be applied, either as part of the programme or externally with the aid of a desk calculating machine. Again, where a differential equation is being integrated, a very good check may be obtained by repeating the integration with a different interval; a similar remark applies to the approximate evaluation of a definite integral. It should be noted that the programmer would in any case probably decide to make use of these methods of checking in order to control rounding-off and truncation errors, which can produce effects very similar to those produced by sporadic malfunctionings of the machine. Thus, if overall checks of this kind can be used, it does not greatly add to the load on the programmer if he is made responsible for the control of machine errors as well.

In matrix manipulation use can often be made of an extra column for checking purposes. Some caution, however, must be exercised in taking over techniques of this kind from ordinary computing practice, since in an automatic computing machine a check on the working of

the store is necessary, whereas, when using paper and pencil, it may be assumed that a number once written will not change its value.

Iterative methods have the advantage in automatic, as in hand, computing that, provided the process converges, sporadic errors do not affect the validity of the result. Moreover, a good check both on the performance of the machine and on the correctness of the programme can be obtained by examining the last few convergents.

In view of the considerations which have been set forth above, I feel that, with the exception of circuits for checking the action of the printer or other output equipment, built-in checking devices are of doubtful utility if viewed as devices for giving the programmer confidence in the results. They can, however, serve a very valuable function in enabling the maintenance engineer to locate faults with certainty and with a minimum of delay. From this point of view they are properly regarded as items of test gear. It is desirable that checking circuits introduced with this end in view should not add greatly to the bulk of the machine, or, if this is unavoidable, that the design should be so arranged that faults occurring in the checking circuits themselves should not render the machine unserviceable. Otherwise the presence of checking circuits may result in the occurrence of more breakdowns in the machine than would otherwise be the case.

Every designer of a computing machine must decide for himself what checking devices, if any, to incorporate. I do not think it will ever be possible to lay down a policy applicable to all circumstances. The main conflict will always be between keeping the machine as simple and small as possible and in making it easy to locate faults. Advances in the techniques of constructing and maintaining machines will make it less necessary to keep equipment to a minimum; on the other hand, by improving serviceability, they will also make the provision of elaborate fault-finding equipment less necessary.

THE SPEED OF ELECTRONIC MACHINES

Electronic machines vary a great deal in speed. For example, the EDSAC takes about $1\frac{1}{2}$ msec to execute each order (4 msec for multiplication), whereas the machine at the Institute for Advanced Study, Princeton, takes only 60 μsec for addition (700 μsec for a multiplication). These machines have very similar order codes, and the comparison made gives a true impression of the relative speeds of the machines, as far as the execution of a programme of orders contained entirely within the high-speed store is concerned. It is not, however, always as easy as this to compare two machines, since the facilities provided for the use of the programmer, and even the principles on which programmes are constructed, may be quite different. In particular, the basic pulse interval used in a machine is, taken alone, a very poor guide as to its speed, and account should also be taken of

the number of parallel channels used to handle information and the access time of the store. Also, in electronic machines of the EDVAC type, a good many of the orders in the programme are concerned with the organization of the calculation—in fact with the programme itself —rather than with furthering the computation in hand; for example, in the programme for summing a sequence of numbers given on p. 47, only about a quarter of the orders are concerned with the arithmetic, the others being used to modify addresses in other orders and to keep count of the number of times the cycle has been repeated. If sufficient storage space is available, programmes can often be speeded up by writing the orders out at length instead of using a repetitive cycle; very nearly as much speeding up can be obtained by the use of B-registers if these are available.

The above considerations refer to the speed with which a programme wholly contained within the high-speed store is executed. The total time taken for a large-scale computation, however, depends also on the speed of input and output and on the speed with which transfers can be made to and from the auxiliary store; it also depends on the suitability for the particular job in hand of the input, output, and auxiliary storage devices fitted to the machine. It may easily happen that a machine which is fast when used on one job may be comparatively slow when used on another, whereas another machine, of similar general capabilities, may be slow for the first job and fast for the second.

CHAPTER 3

THE PRINCIPLES OF PROGRAMME CONSTRUCTION

An example was given in Chapter 2 to show how a programme might be written to compute the value of a simple algebraic expression, when numerical values were given for the symbols appearing in it. It was also shown how a repetitive cycle, in which a group of orders was repeated a specified number of times, could be formed, and the important technique was illustrated whereby orders which have the effect of modifying other orders can be included in the programme. Reference was also made to the use of subroutines. These are self-contained sequences of orders each of which performs a distinct part of the calculation in hand.

It is not essential to divide a programme into subroutines, but to do so facilitates its construction and makes it easier to comprehend as a whole. Moreover, subroutines for performing commonly recurring operations can be prepared once for all; they can then be kept in a *library*, so that they are available for the use of all programmers working with the machine. It is interesting to note that the word 'library' was first used in this connexion by Babbage. Obvious examples of subroutines suitable for including in a library are ones for calculating sines, cosines, exponentials, and similar functions, but it will be seen later that the range is much wider than this, and includes subroutines for performing certain processes, such as numerical quadrature and the numerical integration of differential equations. Library subroutines can be thoroughly tested before being included in the library, so that a programmer incorporating one in his programme can be morally sure that it is free from errors. This may not seem an important point to anyone without experience in writing programmes, but in practice it is found that the detection and elimination of all errors is one of the main preoccupations of the programmer; it is the rule rather than the exception for a newly written programme to contain one or more 'slips' or 'blunders', and more will be said later about this subject.

Another advantage of constructing programmes from subroutines is that it is then comparatively simple to make alterations found to be necessary as the work proceeds; all that is usually necessary is to modify one or two of the subroutines or to replace them by new ones. Even in cases where the modifications required are more extensive, it will very likely be found that some of the subroutines from the old programme can be carried over into the new one.

The greater part of this chapter will consist of a discussion of the

problems encountered in using subroutines systematically for the construction of programmes. The term 'subroutine' will be taken to include any sequence of orders which the programmer finds convenient to treat as a sub-unit of a complete programme, whether it comes from a library or not. Naturally, a programmer will wish to use library subroutines when suitable ones are available, in order to save himself trouble; beyond this, however, the division of a programme into subroutines is a device with no other purpose than to facilitate the work of the programmer, and he is the sole judge of what is appropriate in given circumstances. I shall discuss first the construction of subroutines in general and their assembly into programmes, and later pay attention to the special considerations which arise in the case of subroutines intended for inclusion in a library.

Although modern electronic machines differ to some extent in their order codes, very similar devices and expedients are available in all machines for the use of the programmer. Unfortunately, there are wide differences in the way in which orders are written before being presented to the machine. However, anyone familiar with the method of programming used for one machine will find that, once he has made the sometimes not inconsiderable effort of mastering the notation used for another machine, he is again on familiar ground. This is true even if he has been used to a machine with a single-address code and the new machine has a multi-address code, or vice-versa.

In the greater part of this chapter the notation used for orders will be the same as that used for the EDSAC, but slightly modified and simplified; in particular, the complications which result from the use of both long and short numbers will be avoided by assuming that all numbers and orders are of the same length, namely 17 binary digits (including, in the case of numbers, the sign digit). The functions of orders are indicated by capital letters, some of which suggest the functions they denote. Addresses are written as decimal numbers, and the reader will find this notation easier to follow than notations in which addresses are written in unfamiliar scales, for example, in the scales of 8, 16, or 32. Although differences in the way in which orders are written constitute superficial, rather than fundamental, differences between methods of programming, they are not without importance, especially for the newcomer to the subject. Some remarks on the use of multi-address codes will be made later in the chapter.

The orders which will be used in the examples given in this chapter are as follows:

A *n* F Add the number in storage location *n* into the accumulator.
S *n* F Subtract the number in storage location *n* from the accumulator.
U *n* F Copy the 17 most significant digits of the number in the accumulator into storage location *n*, and retain the whole number in the accumulator.

T *n* F Transfer the 17 most significant digits of the number in the accumulator to storage location *n* and clear the accumulator.
H *n* F Replace the number in the multiplier register by that in storage location *n*.
V *n* F Multiply the number in storage location *n* by that in the multiplier register and add the product into the accumulator.
C *n* F Collate the number in storage location *n* with that in the multiplier register; that is add a 1 into the accumulator in those (and *only* those) digital positions in which both the numbers have a 1.
M *n* F Clear the six most significant binary digits of the accumulator and then add the number in storage location *n* into the accumulator.
R *n* F Shift the number in the accumulator *n* places to the right; that is divide it by 2^n.
L *n* F Shift the number in the accumulator *n* places to the left; that is multiply it by 2^n.
F *n* F Take the order in storage location *n* as the next order to be obeyed.
E *n* F If the number in the accumulator is positive or zero take the order in storage location *n* as the next order; otherwise proceed serially.
G *n* F If the number in the accumulator is negative take the order in storage location *n* as the next order; otherwise proceed serially.
I *n* F Place in storage location *n* the number $b.2^{-16}$ where *b* is the integer represented by the row of holes read from the input tape. Advance the tape to the next row of holes.
Z F Stop the machine.

In the order code given here provision is made for clearing the accumulator (if required) when its contents have just been transferred to the store; in many machines provision is made instead for clearing the accumulator when information is being transferred into it, an order, sometimes called a 'clear and add' order, being provided for this purpose. The practice of clearing locations in the store automatically immediately before new information is transferred into them is general. Rounding-off—not covered in the selection of orders given above—is a distinct operation in the EDSAC, but in most other machines it is combined with some other operation, usually multiplication. For simplicity rounding-off will be omitted in the examples given below. It will also be assumed that the accumulator is clear initially.

THE USE OF SUBROUTINES

The discussion of subroutines which follows will be illustrated by reference to a subroutine for calculating the sum of the cross products of two series of numbers, $a_1, a_2, \ldots, a_{50}$ and $b_1, b_2, \ldots, b_{50}$, namely

$$S = a_1b_1 + a_2b_2 + \ldots + a_{50}b_{50};$$

S may be otherwise described as the scalar product of two vectors in 50 dimensions. This subroutine is chosen not because it is of particular importance or of wide utility, but because it is simple and enables a number of separate points to be illustrated. It is assumed that $a_1, a_2, \ldots, a_{50}$ are in storage locations 100, 101, . . ., 149, and that $b_1, b_2, \ldots, b_{50}$ are in storage locations 300, 301, . . ., 349. A sub-

routine for calculating S and placing it in storage location 0 is as follows:

Location in store	Orders	
10	T 0 F	clear 0
11	S 29 F	part of counting sequence (see p. 46)
25→ 12	T 28 F	
13	A 0 F	add product to C(0)
14	(H 100 F)	
15	(V 300 F)	
16	T 0 F	
17	A 27 F	modify orders in 14 and 15
18	A 14 F	
19	T 14 F	
20	A 27 F	
21	A 15 F	
22	T 15 F	
23	A 28 F	part of counting sequence
24	A 27 F	
25	G 12 F	
26	F . . . F	jump to next part of programme
	Numbers	
27	2^{-15}	
28	—	used for counter
29	50.2^{-15}	

In this example the orders in 14 and 15 are shown in brackets in order to draw attention to the fact that they are altered by other orders in the programme. A similar convention will be adopted throughout this chapter. The brackets are ignored when punching the input tape for the machine.

If the subroutine is required for use a second time the orders in 14 and 15 must be reset to their original form. This is done automatically in the following subroutine, which also effects a slight economy in storage space occupied by making use of a modified system of counting in which one of the variable orders is itself used as the counter. Note that the numbers have been written at the end of the sequence in the same notation as that used for orders. This is possible, since orders are represented inside the machine as numbers, so that, if convenient, there is no reason why numbers should not be represented outside the machine as orders; they are then described as *pseudo-orders*. It will frequently be found that constants needed for modifying orders can most conveniently be represented in this way. It is convenient to indicate pseudo-orders in a programme by means of a vertical double line as used in the following example. Note that $p = 0$ so that P n F stands for the number $n.2^{-15}$.

10	T	0 F	clear 0
11	A	28 F	set variable orders
12	U	17 F	
13	M	29 F	
14	T	18 F	

```
26→15    T   4 F    clear accumulator (4 is used as 'rubbish bin')
   16    A   0 F  ┐
   17    (      ) ├ add product to partial sum; variable orders become
   18    (      ) ┘   H 100 F, V 300 F on first repetition
   19    T   0 F  ┘
   20    A  17 F  ┐
   21    A  30 F  │
   22    U  17 F  ├ modify variable orders
   23    M  29 F  │
   24    U  18 F  ┘
   25    S  31 F  ┐ test for end
   26    G  15 F  ┘
   27    F  ...F    jump to next part of programme
   28  ‖ H 100 F  ┐
   29  ‖ V 200 F  │
   30  ‖ P   1 F  ├ pseudo-orders
   31  ‖ V 350 F  ┘
```

Note that the M order is here used when the function digits of an order are to be modified. In practice, this is more convenient than the use of the method illustrated on p. 47, since account need not be taken of the numerical values ascribed to the function letters. The variable orders are shown blank, since they are set by the subroutine. In practice, however, dummy orders must be punched on the tape in their place, so that the numbering of subsequent orders shall be correct. Since these dummy orders are overwritten by the programme, it does not matter what form they take.

The number of orders in the above subroutine may be further slightly reduced by combining the orders which set the variable orders to their initial values, and those which modify them and test for the end of the calculation. For the present purpose, however, this refinement will not be necessary.

Relative Numbering of Addresses

In order to achieve a flexible system whereby subroutines may be combined to form programmes intended to be placed as a whole in the high-speed store, it is necessary that subroutines should be drawn up so that their correct operation does not depend on their being placed in a particular part of the store. That this is not necessarily a trivial point may be seen by examining the subroutine for calculating a scalar product which has just been given. It will be seen that the addresses in orders in locations 11–14 and 20–26 are all those of orders or pseudo-orders in the same sequence, and therefore depend on where the sequence as a whole is placed in the store; for example, if the first order were placed in storage location 20 instead of 10, the addresses in all the orders just mentioned would have to be increased by 10.

If the order code of the machine provides for addresses to be specified relatively as well as absolutely, the subroutine can be written in

such a form that no modification is necessary. Examples of the orders which are needed for this to be possible are:

A′ *n* F Add the content of storage location $m+n$ into the accumulator, where m is the number of the storage location containing this order.

G′ *n* F If the number in the accumulator is negative, send control to the order in storage location $m+n$, where m is the number of the storage location containing this order; otherwise, proceed to the next order in sequence.

It is assumed that n can have negative as well as positive values.

However, the EDSAC and many other machines are not equipped with facilities for the use of relative addresses, and it is necessary, when using these machines, that addresses which depend on the placing in the store of the subroutine containing them should be adjusted before the subroutine is used. The method introduced at the beginning of 1950 for doing this in the case of the EDSAC will now be described. The subroutines are stored in the form of short lengths of punched paper tape, and, when required for use, are copied mechanically on to programme tapes; these programme tapes can then be read directly into the high-speed store. The reading of a tape is carried out under the control of an initial input routine in the manner described on pp. 58–61, but, in addition to the functions there described, the initial input routine carries out the necessary adjustment of orders whose addresses depend on where they are placed in the store.

The subroutine for forming a scalar product is punched in the following form:

Relative location of order	G K		
0	T	0 F	clear 0
1	A	18 *θ*	set variable orders
2	U	7 *θ*	
3	M	19 *θ*	
4	T	8 *θ*	
16→5	T	4 F	clear accumulator (4 is used as 'rubbish bin')
6	A	0 F	add product to partial sum; variable orders become H 100 F, V 300 F on first repetition
7	(	)	
8	(	)	
9	T	0 F	
10	A	7 *θ*	modify variable orders
11	A	20 *θ*	
12	U	7 *θ*	
13	M	19 *θ*	
14	U	8 *θ*	
15	S	21 *θ*	test for end
16	G	5 *θ*	
17	F	... F	jump to next part of programme
18	H	100 F	pseudo-orders
19	V	200 F	
20	P	1 F	
21	V	350 F	

Here the letters G K form a *control combination*; they serve the purpose of a marker on the tape and cause a record to be made of the storage location in which the next order read from the tape will be placed; that is, they cause the number forming the address specified in the 'transfer order' (see p. 61) to be copied into a certain storage location set aside for that purpose. Any orders, subsequently read from the tape, which are terminated by the code letter θ have this number added to them. Thus, if the subroutine were placed in the store with its first order in storage location 10, all addresses terminated by θ would have 10 added to them before they reached their final locations in the store. This system, in effect, provides the programmer with a system of relative numbering for orders within a subroutine.

The initial input routine used with the EDSAC also provides facilities whereby the programmer can number other sequences of orders and numbers, not necessarily belonging to a subroutine, relatively to their first member. Each independent sequence—up to twelve are possible—is distinguished by one of the code letters ϕ, H, N, M, &c., and any address terminated by one of these code letters has the content of a certain storage location, different for each code letter, added to it. By placing suitable numbers in these storage locations before the sequences are read from the input tape, the programmer can cause the relative addresses to be replaced by the correct absolute addresses. The following table will make this more explicit by giving the numbers of the storage locations corresponding to the various code letters. The table also gives the numerical values associated with the letters, and it will be seen that from ϕ onwards these are in numerical order.

Code letter	Numerical value	Storage location whose content is added to address
θ	18	42
ϕ	20	44
H	21	45
N	22	46
M	23	47
Δ	24	48
.	.	.
.	.	.
.	.	.
V	31	55

Thus, if storage location 45 (for example) contains h, an order punched on the tape as A 100 H is equivalent to A $100+h$ F.

It is frequently convenient to number, relatively to their first members, the sequences of constants and pseudo-orders associated with a

programme or subroutine; for example, the subroutine just given may be rewritten as follows:

```
                G K
        0       T    0 F
        1       A    0 H
        2       U    7 θ
        3       M    1 H
        4       T    8 θ
16 →    5       T    4 F
        6       A    0 F
        7       (         )
        8       (         )
        9       T    0 F
       10       A    7 θ
       11       A    2 H
       12       U    7 θ
       13       M    1 H
       14       U    8 θ
       15       S    3 H
       16       G    5 θ
       17       F ... F
H      0    ‖  H  100 F
       1    ‖  V  200 F
       2    ‖  P    1 F
       3    ‖  V  350 F
```

The Initial Input Routine

In this section (which the reader may omit if he so desires) an initial input routine is given for performing the functions which have just been described. In addition to responding to the control combination G K, this initial input routine responds to the control combinations T *m* K and E *m* K P F, where *m* ($0 \leqslant m \leqslant 1023$) stands for any address. T *m* K sets the address in the 'transfer order' so that subsequent orders read from the tape are placed in storage locations *m*, *m* + 1, &c. E *m* K P F is punched at the end of the orders on the tape and causes control to be transferred (with the accumulator clear) to storage location *m*, which is assumed to contain the first order of the programme. It will be observed that all control combinations are terminated by K; this letter has a numerical value of 14, which is intermediate between the values of decimal digits and the numerical values of the code letters used to terminate orders. Control combinations are, therefore, readily distinguishable from orders.

The reader should be able to follow the action of the initial input routine with the aid of the flow diagram given in Fig. 3·1. He will observe that all code letters terminating orders, including F, cause the content of some storage location to be added; however, the location corresponding to F (namely location 41) always contains zero, so that addresses terminated by F are not modified. The routine is as follows:

```
25┐  0     H  40 F   10/16 to multiplier register
28├→ 1     T  58 F   clear 58
38┘  2     I  59 F ┐
     3     A  59 F ├ read, shift, and store function letter
     4     L  12 F ┤
     5     T  39 F ┘

17→  6     I  59 F ┐
     7     A  59 F ├ read, shift, and store next character x
     8     L   1 F ┤
     9     U  59 F ┘

    10     S  56 F   test next character
    11     E  18 F   jump if not a figure, i.e., if x ⩾ 10
    12     T  62 F   clear accumulator

    13     V  58 F ┐
    14     L   4 F ├ decimal-binary conversion
    15     A  59 F ┤
    16     T  58 F ┘

    17     F   6 F
           ------
11→ 18     S  60 F
    19     E  29 F   jump if not K, i.e., if x ⩾ 15
    20     T  62 F   clear accumulator

    21     A  39 F ┐ test function letter; jump if G
    22     G  26 F ┘

    23     A  58 F ┐ form and plant new order, to replace 'transfer
    24     T  34 F ┘   order'

    25     F   1 F
           ------
22→ 26     A  34 F ┐ copy address in 'transfer order' into 42 (note that
    27     T  42 F ┘   g + t = 0)

    28     F   1 F
           ------
19→ 29     A  61 F ┐ form and plant order for adding content of
    30     T  33 F ┘   specified location to address

    31     A  39 F ┐
    32     A  58 F ├ assemble order; order in 33 is planted by 30
    33     (     ) ┘

    34     (     )   'transfer order'

    35     A  34 F ┐
    36     A  57 F ├ increase address of 'transfer order' by unity
    37     T  34 F ┘

    38     F   1 F
           ------
    39     (     )   used for temporary storage of function letter
    40  || J     F   pseudo-order ≡ 10/16

    41             ┐
     .             │
     .             ├ 15 locations referred to by code letters
     .             │
    55             ┘

    56  || P  10 F
    57  || P   1 F
    58     (     )   used to build up address
    59     (     )   used for temporary storage
    60  || P   5 F
    61  || A  39 F
    62     (     )   'rubbish bin'
```

The initial input routine used with the EDSAC is similar to that just given, but provides certain extra facilities. In addition, when it was being constructed, great stress was placed on compressing it into as small a number of storage locations as possible, and a number of unorthodox tricks were used to this end; it is therefore less easy to understand than the initial input routine given above. The principal extra facilities relate to the use of long and short numbers. The code letter D (numerical value 19) causes the content of storage location 43 to be added to an order; this location always contains 2^{-16}. In addition, the symbol π punched before a code letter causes the addition

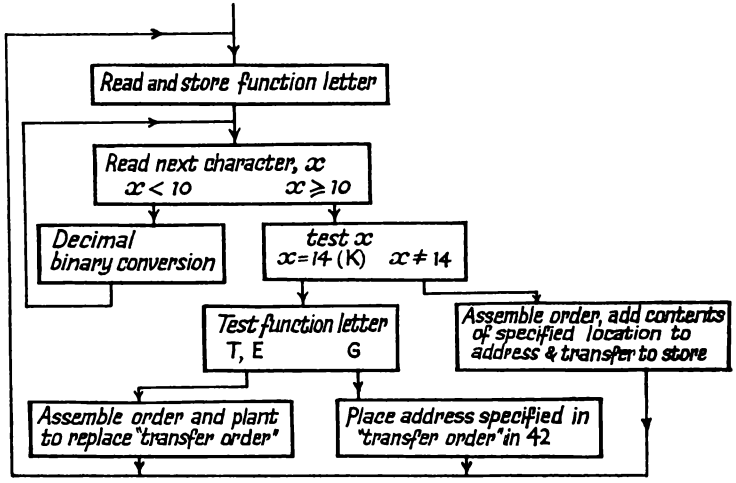

Read and store function letter
Read next character, x
x < 10
x ≥ 10
Decimal binary conversion
test x
x = 14 (K)
x ≠ 14
Test function letter
T, E
G
Assemble order, add contents of specified location to address & transfer to store
Assemble order and plant to replace "transfer order"
Place address specified in "transfer order" in 42

Fig. 3·1.—Flow diagram for initial input routine.

of 2^{-16}, so that an address terminated by a code letter can be made to refer to a long number. The reader who is interested will find the initial input routine used in the EDSAC given in full in references 89 and 98.

The idea lying behind the system which has just been described is that subroutines, or other sequences of orders or numbers punched on a paper tape, can be converted to a new form on being transferred to their final locations in the high-speed store. One might express this by saying that the code used to represent orders outside the machine differs from that used inside, the differences being dictated by the different requirements of the programmer on the one hand, and of the control circuits of the machine on the other. The principle of conversion during input is capable of wide application, and I shall return to it later in the chapter.

If the library of subroutines is stored on a magnetic tape, or in some other kind of auxiliary store permanently accessible to the

machine, instead of in the form of lengths of punched paper tape, the modification of addresses must take place when subroutines are transferred to the high-speed store. One method of modifying addresses which refer to orders or pseudo-orders within the same subroutine (orders which in the EDSAC system would be terminated by the code letter θ) was proposed by von Neumann and Goldstine in 1948. The subroutine is first transferred from the auxiliary store to the place it is to occupy in the high-speed store. Control is then sent to a processing routine which extracts each order of the subroutine in turn, examines its address, and modifies it if it is found to refer to an order or pseudo-order within the subroutine. This treatment must be applied only to orders and not to pseudo-orders, and it is therefore convenient for the pseudo-orders to be grouped together in a block at the end of the subroutine. Before control is transferred to the processing subroutine, the number of the location holding the first order of the subroutine and the number of orders to be processed must be placed in locations where they will be available for use by the processing routine. Since the orders to be modified are selected by testing their addresses to see whether they lie in a certain range, some restrictions are necessary on the use of absolute addresses when drawing up subroutines. Also, special arrangements must be made to deal with shift orders if these are included in the order code of the machine, since the address sections of such orders contain, instead of an address, a specification of the number of places through which shifting is to take place. A further limitation of the method is that it cannot be extended to provide relative numbering facilities of the kind associated with the code letters ϕ, H, N, &c., in the EDSAC.

An alternative method of indicating to the processing subroutine which orders contain addresses that must be modified is to include a list, or directory, of these orders at the end of the subroutine. Since one binary digit only is required to indicate whether a given order is to be modified or not, a single number can carry information relating to as many orders as there are binary digits in the number. For example, the number 011000101 . . . could indicate that the second, third, seventh, ninth . . ., orders were to be modified. Another method would be to use two storage locations in the auxiliary store to contain each order, the first containing the order itself and the second containing tags which indicate what modifications are necessary. These tags would correspond to the code letters punched on the paper tape after the addresses in the EDSAC system. A subroutine when first transferred to the high-speed store would occupy twice as much space as it would finally need when the adjustments had been carried out by the processing subroutine. This system would be rather wasteful of space in the auxiliary store, a disadvantage which would be less serious with magnetic tape than with a magnetic drum.

The Incorporation of Subroutines in Programmes

Three desiderata need to be borne in mind when designing or evaluating a system whereby a collection of library or other subroutines may be combined together to form a complete programme. First, when the programmer is concerning himself with the overall strategy of the calculation he should be able to think of the subroutines as self-contained entities, and it should not be necessary for him to concern himself with their mode of operation or with the number of orders they contain. Secondly, repeated use of a subroutine at various points in the programme should be facilitated and it should be unnecessary for duplicate copies of the same subroutine to be placed in the high-speed store. Thirdly, the number of orders introduced for the sole purpose of joining together the various parts of the programme should be as small as possible. In what follows it will be assumed at first that the entire programme can be accommodated in the high-speed store.

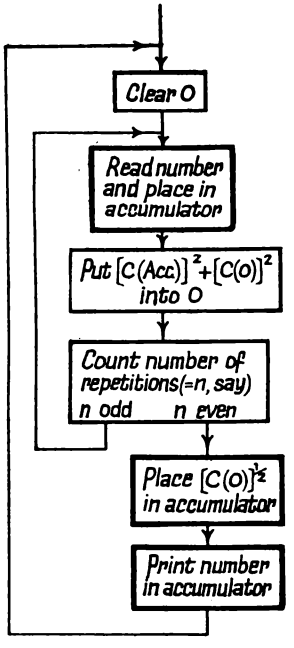


Fig. 3·2.—Flow diagram for a simple calculation.

In some cases the assembly of the subroutines to form the complete programme may be a very simple matter; it may, for example, be necessary only to place the subroutines end to end and to follow them with suitable counting and transfer of control orders to cause the programme to be repeated the required number of times. In other cases, certain subroutines may need to be operated a number of times for each repetition of the programme, or control may need to be sent from one part of the programme to another according to specified criteria. In such cases the preparation of a flow diagram may facilitate the construction of the programme. A simple example is given in Fig. 3·2; this shows a flow diagram for a programme which causes pairs of numbers to be read into the machine and the square root of the sum of their squares to be printed. This programme has one main loop and one subsidiary loop. The boxes bounded by thick lines represent subroutines, and those bounded by thin lines represent sequences of orders introduced to connect the subroutines together. If a flow diagram or some equivalent device is not used, it is difficult

to keep clear in one's mind the action of the programme, since the various parts become so welded together that they lose their identity. In practical problems flow diagrams can become very complicated, and much ingenuity is needed in their construction.

A considerable simplification can be effected—especially when it is necessary for subroutines to be operated more than once—by basing the programme on a *master routine*. This is a continuous sequence of orders which performs all the calculation, counting, and manipulation which is not covered by one or other of the subroutines. When a point is reached in the master routine at which one of the subroutines must be operated, control is sent to that subroutine in such a way that, when the subroutine has done its work, control will return to the master routine. A convenient way of doing this, which makes it unnecessary for the programmer to concern himself with the details of the process by which control is returned, was devised by Wheeler. The subroutines are specially constructed for the purpose and contain orders for forming and planting within themselves the necessary *link orders* for returning control to the master routine. Such a subroutine is called a *closed subroutine*, and provided that it is called in in the correct manner, the action of returning control to the master routine is quite automatic. A closed subroutine can be placed anywhere in the store at the discretion of the programmer. The term *open subroutine* is used to describe a subroutine designed for embodying as a whole within a programme. Control enters an open subroutine at the first order and, when the subroutine has done its work, passes to the orders of the programme which come immediately afterwards.

Two actions are necessary for calling in a closed subroutine; a record must be made of the place reached in the master routine, and control must be sent to the subroutine. In the system adopted for use with the EDSAC the following two orders are used for calling in a closed subroutine:

m	A m F
$m+1$	F n F

Here it is supposed that the first order of the subroutine is placed in storage location n. It will be seen that the address specified in the first order is the same as that of the location in which the order is placed; this order thus has the effect of adding itself into the accumulator. The next order gives an unconditional transfer of control to the first order of the subroutine.* Thus, when control enters a sub-

* When the system was first introduced in 1950 the EDSAC had no order giving an unconditional transfer of control. The order G n F was therefore used instead. This had the same effect, since an A order has a negative form inside the machine. For a similar reason there was a small difference in the pairs of orders used for planting the link inside the subroutine; A 3 F was used instead of M 3 F (no M order being then available) with C(3) = U 2 F.

routine a record exists in the accumulator of the place reached in the master routine. The orders in the subroutine which form and plant the link order, which will return control in due course to the master routine, are given below. Note that C(3) = F 2 F.

n	M 3 F	} form and plant link order
$n+1$	T $n+p$ F	
$n+2$	...	} operational orders of subroutine
...	...	
...	...	
$n+p$	( )	becomes link order

When control enters the subroutine, the accumulator contains A m F. The first order of the subroutine modifies this by replacing the function letter A by F and adding 2 to the address. The resulting order, F $n+2$ F, is then planted at the end of the subroutine, where it serves as a link order to transfer control back to the master routine at a point immediately after the pair of orders calling in the subroutine.

One great advantage which the programmer gains by using closed subroutines is that, when writing the master routine, he is enabled to concentrate on the main task in hand without having to bother about all the details of the calculation. Once he has decided to find room in the store for a subroutine for performing some particular operation it is to all intents and purposes as though the basic order code of the machine covered that operation. This is illustrated by the following example, in which a programme is given for reading pairs of numbers from the input tape and printing the square root of the sum of the squares of each pair. This example has already been used to illustrate the use of flow diagrams. The following subroutines are required and it is assumed that they are placed in the store with their first orders in the storage locations indicated:

Function of subroutine	Reference letter	Location of first order
Read number from tape and place in storage location 0	*R*	100
Replace number in storage location 0 by its square root	*S*	150
Print the number in storage location 0 . .	*P*	200

The master routine is then as follows (the two columns on the right explain the action in detailed and in general terms respectively):

Location of order in store	Order		
25→10	A 10 F	} call in *R*	read x from tape into 0
11	F 100 F		
12	A 0 F	add x into Acc.	} put x in 4
13	T 4 F	transfer x to 4	
14	A 14 F	} call in *R*	read y from tape into 0
15	F 100 F		

Location of order in store	Order			
16	H	4 F	x to multiplier register	form x^2+y^2 and place in 0
17	V	4 F	add x^2 into Acc.	
18	H	0 F	y to multiplier register	
19	V	0 F	add y^2 into Acc.	
20	T	0 F	transfer x^2+y^2 to 0	
21	A	21 F	call in S	replace x^2+y^2 in 0 by $\sqrt{(x^2+y^2)}$
22	F	150 F		
23	A	23 F	call in P	print $\sqrt{(x^2+y^2)}$
24	F	200 F		
25	F	10 F	repeat	

When a subroutine has to be called in at more than one point in the master routine, the method of forming and planting the link order described here is highly efficient as regards the number of orders it requires. When a subroutine is required to be operated at one point only in the master routine, the use of a closed subroutine instead of an open one makes the programme longer by five orders. The extra computing time taken is hardly a matter of importance in a high-speed machine, but the extra storage space occupied may be a disadvantage. However, the great convenience which the use of closed subroutines offers to the programmer has led to their general adoption at Cambridge, not only for the library, but sometimes also for parts of a programme which have to be constructed by the programmer himself. Having constructed a closed subroutine—and perhaps tested it by means of a short master routine specially constructed for the purpose—the programmer then forgets about its details and treats it as one of the blocks out of which to build the whole programme.

Other systems on which closed subroutines might be operated will occur to the reader. For example, the following orders might be used in the master routine:

m	A $m+2$ F
$m+1$	F n F
$m+2$	F $m+3$ F

In this case, the subroutine need contain only a T order to plant the link order at the end of the subroutine, since the link order exists ready-made in the accumulator when the subroutine is entered. This method has the disadvantage that one extra order is needed in the master routine and that the programmer is put to slightly more trouble than with the other system. If the machine is provided with B-registers it is convenient to use one of these to store the information needed for returning control to the master routine.

In some machines an order is provided which gives the programmer access to the number in the sequence control register, and the device of using an order which adds itself into the accumulator for recording

the position reached in the master routine is unnecessary. There is no reason why the same order should not cause control to be transferred to the subroutine. One order only is then necessary for calling in a closed subroutine. The designers of Whirlwind I at the Massachusetts Institute of Technology provided an order of this kind; it is written *sp n* and causes control to be sent to storage location *n* with the former content of the sequence control register in one of the registers of the arithmetic unit, from which it can be extracted by an order in the subroutine.

Although in the discussion given above reference has always been made to a subroutine being called in from the master routine, it frequently happens that one subroutine calls in another. Indeed, programmes with five or six 'levels' of subroutines are not uncommon. It is important, therefore, that any method of handling closed subroutines adopted for general use should permit the programmer to use as many levels of subroutines as he may require. Simple systems, in which a subroutine always returns control to a fixed point in the store when it has finished its work, usually offend in this respect. The system described here not only enables an unlimited number of levels to be used, but it also enables a given subroutine to be used on more than one level; for example, a subroutine for extracting square roots can be called in both from the master routine and from any of the other subroutines which happen to require it.

Many of the devices which have been described in connexion with subroutines are equally useful to the programmer when writing a master routine. For example, it is convenient to write a master routine in such a form that it can be placed in any part of the store that afterwards turns out to be convenient, the code letter θ being made use of for this purpose. Similarly, other code letters can be used to define relative numbering systems for the sequences of constants and pseudo-orders referred to in the master routine. Structurally, therefore, there is little to distinguish the master routine from one of the subroutines; its special status arises from the fact that it is concerned with the organization of the programme at the highest level.

An important application of the principle whereby one subroutine calls in another is the use of *auxiliary* subroutines in connexion with library subroutines designed to carry out standard processes on arbitrary functions. Consider, for example, the evaluation of a root of the equation $F(x) = 0$ by means of the rule of false position, or some other algorithm, given an approximate value of the root as a starting point. At first sight it would seem to be impossible to write a subroutine for this purpose unless the form of $F(x)$ were specified. Such a subroutine may be written, however, if it is assumed that the computation of $F(x)$ is performed by means of a second, or auxiliary, subroutine. This subroutine may conveniently be of the closed type and

be so designed that, when it is called in, it replaces a value of x, placed beforehand in storage location 0, by $F(x)$. If the existence of such an auxiliary subroutine is assumed, all the writer of the main subroutine need do, when he reaches a point at which a value of $F(x)$ is required, is to write down orders for placing the value of x in storage location 0 and for calling in the auxiliary subroutine.

Subroutines may be constructed on this principle for numerical quadrature, for the solution of ordinary differential equations, and for various other purposes in addition to the one mentioned above. In all cases, when such a subroutine is taken from the library, it is necessary for the user himself to construct a suitable auxiliary to define the functions occurring in his particular application.

It is sometimes convenient to design a subroutine with one or more adjustable parameters which can be given different values each time the subroutine is called in. These are known as *programme parameters*. One situation in which a programme parameter is useful occurs when an output subroutine is required to print results rounded off to different numbers of significant figures on different occasions. Another occurs in the case of an interpolating subroutine which, on one occasion, is required to interpolate into a table with its first entry in a certain location in the store and, on another occasion, to interpolate into a table with its first entry in a different location.

The reader will be able to think of a number of ways of providing the facilities just described. For example, if the machine has B-registers, programme parameters may very conveniently be placed in them before the subroutine is called in. In the system used with the EDSAC the programme parameters are written immediately after the orders calling in the subroutine. During operation of the subroutine they are picked up by orders manufactured for the purpose and planted at appropriate points within the subroutine. The link order must be such as to send control to the order in the master routine which comes immediately after the programme parameters. This method of handling programme parameters is very convenient for the programmer, although it makes the construction of the subroutine rather complicated. Since the values of programme parameters are not substituted into a subroutine until it is about to be operated, they need not be fixed in advance by the programmer, but can, if necessary, be computed by the programme itself.

Interpretive Subroutines

It was pointed out on p. 99 that the use of closed subroutines in effect provides the programmer with additional 'orders' which supplement those in the basic order code of the machine. It will now be shown how an *interpretive* subroutine can be used to enable the programmer to work with a completely different set of orders from those

built into the machine. Each order in the programme written in the new code is interpreted by the subroutine and the requisite sequence of basic machine operations carried out.

As an example of an interpretive subroutine, one which facilitates the addition and subtraction of complex quantities will be described. It will be assumed that the real and imaginary parts of each complex number are held in consecutive short storage locations. Thus if $z = x + iy$ and x is in storage location n, y will be in storage location $n+1$; for short, the complex number z may be said to be in storage location n. Two storage locations, namely 10 and 11, will be set aside to act as a *complex accumulator* which will play the same role in the addition and subtraction of complex numbers as the machine accumulator plays in ordinary additions and subtractions. The subroutine is designed to enable the programmer to use the following order code, which, it will be seen, resembles closely the ordinary EDSAC code for handling real numbers.

A *n* F add the complex number in storage location *n* into the complex accumulator.
S *n* F subtract the complex number in storage location *n* from the complex accumulator.
T *n* F transfer the complex number in the complex accumulator to storage location *n*, and clear the complex accumulator.
U *n* F transfer the complex number in the complex accumulator to storage location *n*, but do not clear the complex accumulator.

Orders in this code will be called *interpretive orders*.

When the programmer wishes to make use of the interpretive subroutine, he calls it in by means of a pair of orders similar to those used for calling in an ordinary closed subroutine. He then writes the programme of orders expressed in the interpretive code; this may be of any length and is finally terminated by the (interpretive) order F *n* F. When this order is reached control is returned to the master routine at storage location *n*. From this point onwards (or until the interpretive subroutine is called in again) the programme is written in the ordinary way in terms of machine orders.

The following example shows how a programme would be written for evaluating the expression $a + b - c$, and for placing the result in storage location 56; a, b, and c are complex numbers contained in storage locations 50, 52, and 54. It is assumed that the interpretive subroutine is placed with its first order in storage location 100.

m	A	*m* F	} call in interpretive subroutine	
m+1	F	100 F		
m+2	A	50 F	add *a* into complex accumulator	} interpretive orders
m+3	A	52 F	add *b* into complex accumulator	
m+4	S	54 F	subtract *c* from complex accumulator	
m+5	T	56 F	transfer result to store	
m+6	F	*m*+7 F	revert to ordinary coding	
m+7		...		
H				

The orders in the interpretive subroutine are given below. The reader should be able to follow the way in which the interpretive orders are extracted one by one from the programme and the correct sequence of operations carried out. The annotation applies to the extraction and execution of the first interpretive order in the above example, namely A 50 F.

	G K	
0	A 16 θ	} form and plant A $m+2$ F (accumulator contains A m F on entry)
1	T 2 θ	
15→2	( )	becomes A $m+2$ F; extract A 50 F from the master routine
3	U 7 θ	plant A 50 F
4	A 17 θ	
5	T 10 θ	plant A 51 F
6	A 10 F	
7	( )	becomes A 50 F } add real part into complex accumulator
8	T 10 F	
9	A 11 F	
10	( )	becomes A 51 F } add imaginary part into complex accumulator
11	T 11 F	
12	A 2 θ	} increase address of order which extracts interpretive orders from master routine
13	A 17 θ	
14	T 2 θ	
15	F 2 θ	return control to beginning
16	P 2 F	} constants
17	P 1 F	

In interpretive subroutines which provide for more complicated operations than the one just described, it is usually necessary to separate the function digits and the address digits of each interpretive order and to interpret them separately. For this purpose, use may be made of an order for performing a logical operation, such as the 'collate' order in the EDSAC, which performs the logical operation of 'and'. For example, it will be assumed that the function letters Q, W, E, R, T, which have numerical values 1, 2, 3, 4, and 5, correspond, when they appear in interpretive orders, to five separate operations; in an interpretive subroutine for handling complex quantities these might be addition, subtraction, multiplication, division, and the forming of the modulus. Separate groups of orders for performing these operations must be provided; a sequence of orders—known as a *switching sequence*—designed to send control to the appropriate group is given below. It will be seen that the function digits of the interpretive order are isolated by collating the order with the pseudo-order V 0 F, which has 1s in the digital positions occupied by function digits and 0s elsewhere; the function digits are then shifted to the right and used to form the address of an F order which, when ultimately executed, causes control to be transferred (via a *directory* of F orders) to the appropriate group of orders. The original interpretive order is left in the multiplier register so that the address digits, in their turn, can be

extracted and made use of. When the operation is complete, control returns to the third order of the switching sequence and the process is repeated for the next interpretive order.

	G K	
0	M 16 θ	} form and plant H $m+2$ F
1	T 2 θ	
2	( )	becomes H $m+2$ F
3	A 2 θ	} increase address of order which extracts interpretive order from master routine
4	A 19 θ	
5	T 2 θ	
6	C 17 θ	extract function digits
7	R 11 F	} form and plant switch order
8	A 18 θ	
9	T 10 θ	
10	( )	becomes switch order
11	F α F	} directory; α, β, γ, δ, ϵ are locations of first members of groups of orders corresponding to the function letters Q, W, E, R, T
12	F β F	
13	F γ F	
14	F δ F	
15	F ϵ F	
16	H 2 F	} pseudo-orders
17	V 0 F	
18	F 10 F	
19	P 1 F	

This example shows the way in which orders for performing logical operations are useful when it is desired to separate several distinct pieces of information which have been packed into a single word held in the store. Instead of isolating the digits required in this way, however, it is sometimes more convenient to shift the number to the right or left so that unwanted digits are lost off the end of the register. This could have been done in the example just given.

As an alternative to the system whereby the interpretive subroutine is called in before each group of interpretive orders, it may be arranged that every order in the programme is examined by the interpretive subroutine, and that those orders specially marked (for example, by having a 1 in a digital position which would be otherwise unused) are interpreted, the others being executed as ordinary machine orders. This system gives slightly more flexibility to the programmer, but means that the execution of each machine order takes several times as long as it otherwise would. However, the system has been used successfully in cases in which interpretive orders are in a majority. An example is its use in connexion with Whirlwind I, which has a fundamental number length of 16 binary digits only, so that nearly all computing has to be done by double-length arithmetic. The fact that Whirlwind I is a very fast machine helps to justify the use of this method in certain applications.

An important field in which interpretive subroutines have been found to be of value is the handling of numbers expressed in floating decimal or floating binary form. Indeed, the extent to which one of these latter subroutines can facilitate the writing of programmes for a certain class of problem, by making it unnecessary for the programmer to estimate the range of variation of the numbers occurring in it, has to be experienced to be appreciated. The use of such an interpretive subroutine, however, reduces the computing speed because of the time taken for interpretation; the execution of each interpretive order involves about 30 machine orders. However, the overall slowing down is not as great as this might indicate, since the testing and shifting operations which would otherwise have to be included in the programme to prevent numbers from exceeding capacity or becoming too small, are not required. A fair estimate would be to say that the calculation is slowed down by a factor of five or six. However, in the case of small problems this loss of speed can often be accepted in view of the saving of the time which would otherwise be taken in programming and in eliminating errors from the programme. It should be noted that floating-point operation is particularly useful in those cases in which a good deal of trial and error on the machine —and consequent consumption of machine time—would be necessary before a programme with optimum scale factors could be constructed.

When a binary machine is used, subroutines can be constructed either for floating-binary or floating-decimal operation. Floating-decimal operation is no more difficult to programme and is not appreciably slower in operation. It has the advantage that the results can be printed directly, whereas, if floating-binary operation is used, conversion to floating-decimal form will normally be necessary before printing.

As an alternative to interpretive subroutines, ordinary closed subroutines with multiple entry points can be used to facilitate floating-point and similar operations. The programmer places the numbers to be operated on in fixed locations in the store and then transfers control to the entry point appropriate to the operation he wishes to perform. This method is not quite so convenient for the programmer and makes the master routine appreciably longer. It has, however, the advantage of being faster in operation. The use of interpretive subroutines and similar devices has proved so useful in practice that designers of future machines might do well to consider the provision of some built-in facility designed to speed up their operation.

THE FORMATION OF A LIBRARY OF SUBROUTINES

If a library of subroutines is to be successful as an aid in the construction of programmes, it is essential that the subroutines shall be

designed in such a manner as to be of wide utility. If this is not done the programmer will more often than not find that no subroutine in the library meets his requirements sufficiently closely to be usable. The necessary generality can be obtained by providing the subroutines with parameters whose values can be set by the programmer. These parameters, whose values are fixed once for all in any particular programme, will be called *preset parameters* to distinguish them from the programme parameters described earlier; their main purpose is to make it unnecessary to include in the library a large number of variants, applicable in slightly different circumstances, of each subroutine.

In the EDSAC punched-tape library, preset parameters are handled by means of the code letters H, N, M, &c., which have already been introduced and used for the relative numbering of orders and pseudo-orders. The numerical values to be given to the parameters in any particular programme are punched on the tape ahead of the subroutine and preceded by a control combination which causes them to go into storage locations 45, 46, 47, &c. They are then incorporated in the subroutine by being added in the ordinary way to orders or pseudo-orders terminated by H, N, M, &c. One common use of preset parameters is to fix the locations of numbers to be operated on. For example, the subroutine for forming the scalar products of two vectors could be modified so that the components of the vectors were taken from storage locations a, $a+1$, $a+2$, and $a+200$, $a+201$, $a+202$ and the result put in storage location b by punching it in the following form:

```
          G K
 0        T   0 H
 1        A  18 θ  ⎫
 2        U   7 θ  ⎪
 3        M  19 θ  ⎪ set variable orders
 4        T   8 θ  ⎭
16→5      T   4 F    clear accumulator
 6        A   0 H  ⎫
 7        (       )⎪
 8        (       )⎬ add product to partial sum
 9        T   0 H  ⎭
10        A   7 θ  ⎫
11        A  20 θ  ⎪
12        U   7 θ  ⎬ modify variable orders
13        M  19 θ  ⎪
14        U   8 θ  ⎭
15        S  21 θ  ⎫ test for end
16        G   5 θ  ⎭
17        F ... F    jump to next part of programme
18        H   0 N  ⎫
19        V 200 F  ⎪
20        P   1 F  ⎬ pseudo-orders
21        V 250 N  ⎭
```

Before the subroutine is read into the store the values to be given to the preset parameters (expressed in the forms P *b* F, P *a* F) are placed in storage locations 45 and 46. *b* and *a* are then added during input to the addresses in the orders and pseudo-orders terminated by H and N.

Constants, other than the numbers of storage locations, may be introduced into subroutines by preset parameters in a similar way. For example, the subroutine given on p. 91 could be generalized so as to take the scalar product of two *n*-dimensional vectors, the value of *n* being given by a preset parameter placed in storage location 45. It would only be necessary to replace the pseudo-order in 21 by V 300 H. During input, the value of *n* in storage location 45 would be added to the address of this order and duly incorporated in the subroutine. Orders selected at the discretion of the programmer may be introduced into subroutines in the same way; for example, left- or right-shift orders could be placed as preset parameters in one of the storage locations 45, 46, &c., to be incorporated into the subroutine in place of entries punched as P 0 H, P 0 N, &c., on the tape. This device is useful when it is desired to give the user of a subroutine control over the scale factors used within it. A further use for preset parameters is in integration and similar subroutines, where they may be used to specify the location in the store of the auxiliary subroutine.

One advantage of the method of incorporating preset parameters into subroutines by the use of code letters in the manner just described is that the subroutines, when finally transferred to the high-speed store, are no longer than they would have been if they had been drawn up specially for the particular cases concerned.

Sometimes, the incorporation of a preset parameter involves a piece of manipulation too extensive to be handled by the simple use of code letters. An example occurs in the case of a printing subroutine which is required to round-off the number to *n* decimal places before printing, *n* being given by a preset parameter. The rounding-off is done by adding the number $0{\cdot}5.10^{-n}$ before printing, and this number must be computed (in its binary form) during input and placed in the store along with the subroutine. The computation is performed by means of an *interlude*. The orders necessary to calculate the round-off constant and to place it in its required position in the store (the orders of the interlude) are punched on the subroutine tape along with the orders of the subroutine proper. When the orders of the interlude have been read into the store, reading of the tape is temporarily halted and control is sent to the first order of the interlude. When the round-off constant has been calculated and placed in the store, control is sent back to the initial input routine so that reading of the tape is resumed. The 'transfer order' is first modified so that the orders of the interlude, which have now served their purpose,

are overwritten. The use of interludes enables quite complicated adjustments to be carried out to a subroutine without making the final form of the subroutine any longer than it would have been if specially drawn up to meet the particular circumstances in which it is being used. Sometimes the amount of storage space ultimately occupied by a subroutine depends on the value which has been given to a preset parameter.

Notes on the Construction of Library Subroutines

Subroutines are usually constructed to take the numbers on which they operate from fixed locations in the store; for example, a subroutine for extracting square roots might take the number on which to operate from storage location 0. An alternative system, which is particularly convenient when more than a few numbers are involved, is for the locations from which the numbers are taken to be specified by means of preset parameters. In addition to numbers treated in either of the ways just mentioned, other numbers needed by a subroutine can be expressed by means of programme parameters. This division of numbers into two kinds is introduced purely for the convenience of the programmer. It may be compared to the differentiation between the two arguments n and x of the Bessel function $J_n(x)$, which could just as well be written $J(n, x)$.

Most subroutines require 'working space', that is, a number of locations in which intermediate results may be held. Usually only a few such locations are required, and it is probably most convenient to have the convention that all subroutines use the same ones. When, in the case of an exceptional subroutine, a large number of working locations are required, they can conveniently be specified by means of a preset parameter. Sometimes it is necessary to allocate locations within a subroutine for the purpose of holding intermediate results; for example, a subroutine for quadrature, which calls in an auxiliary subroutine that might destroy any information stored in the conventional working space for subroutines, must have locations within itself set aside for the purpose of accumulating partial sums. Similarly, an output subroutine, which controls the layout of the printed page, must have means within itself for keeping count of the number of columns and lines printed.

Certain constants are frequently required in subroutines and master routines, and there is a case for having these kept permanently in the store. In the system used with the EDSAC provision was made for two such numbers, namely, a pseudo-order used in the formation of link orders by closed subroutines, and the pseudo-order P 1 F; it might be useful also to have P 2 F. If the machine has no shifting order it may be found useful to keep a table of powers of 2 permanently in the high-speed store.

The above considerations indicate that, in the development of a workable and convenient system for writing programmes with the aid of a library of subroutines, some conventions as to the use of storage space are likely to prove useful. However, the number of such conventions should be kept to a minimum in order that the programmer may be left as free as possible in regard to the use he makes of the store. Although this section has been written primarily from the point of view of the construction of library subroutines, similar considerations apply to the construction of subroutines required for individual problems.

Survey of the Contents of a Library of Subroutines

A brief survey will now be given of the subroutines which might form the nucleus of any library.

Basic Subroutines. The subroutines required in this category depend on the particular policy adopted in the design of the machine. In the case of the EDSAC it contains subroutines for reading numbers from the input tape and converting them to binary form, subroutines for converting the results of calculations to decimal form and printing them in the layout required by the programmer, and subroutines for division, the EDSAC having no built-in divider. In the case of other machines some of these subroutines might not be necessary; indeed, a decimal machine equipped with a divider and a method of setting up the layout of the output sheet on a set of switches or a plug board would not need any of them.

The most straightforward form of output subroutine for use in a binary machine is one which takes a number (assumed to be a binary fraction) from the store and prints it as a decimal fraction. It is also convenient, however, to have a subroutine which will print the same number multiplied by a power of 2 just large enough to make the result integral in all cases; for example, in the case of the EDSAC, in which all long numbers are binary fractions containing 34 binary digits, it is convenient to have a subroutine which prints $2^{34}x$, where x is the number in the store. It is similarly convenient to have input subroutines which read decimal integers, divide them by a similar power of 2, and place them in the store. A method giving greater flexibility is to have an input subroutine which reads a number x and places 2^a10^bx in the store, and an output subroutine which prints 2^a10^by, where y is a number in the store. In both cases a and b are parameters. If this system is adopted, two subroutines only, one for input and one for output, are required, although each may be individually slightly longer than the less general subroutines referred to earlier. This might well be no disadvantage in a machine in which subroutines were stored in an auxiliary store and brought into the high-speed store only when required for use.

If the layout of the final printed page is controlled from the programme, as it is in many machines using a teleprinter or an electric typewriter for output, output subroutines giving various kinds of layout are desirable. The simplest prints a single number, followed by a carriage return and line feed, each time it is called in, so that successive numbers appear in a single column. A more elaborate subroutine prints a single number each time it is called in, but contains arrangements for keeping count of the numbers as they are printed and causes the printer to start a new line only after every fifth (say) number has been printed. The numbers then appear in five separate columns. In this case it is also desirable to make arrangements for an extra line feed to occur after a fixed number of lines, so that the lines are divided into blocks. Both the number of columns and the number of lines in a block can be controlled by preset parameters. A further preset parameter can be used to determine the number of decimal digits in each number printed. A further refinement is to arrange means whereby the figures in each number are broken up into groups according to the wishes of the programmer. The insertion of a decimal point can also be provided for if desired. Subroutines for printing integers should preferably have means for suppressing zeros in front of the first significant digit.

Division may be effected by repeated subtraction, but, in the case of most computers, this method is slower than the methods using iterative processes which will be described shortly. The following 'digit-by-digit' method uses a very small amount of storage space, although it is rather slow; it is described here in detail, since similar methods are applicable for other functions. Suppose that we require to calculate a/b, where a and b are both less than 1, and a is less than b. We start with the trial value $\frac{1}{2}$, and multiply this by b. If the result is greater than a, we subtract $\frac{1}{4}$ from the trial value, otherwise we add $\frac{1}{4}$. We now multiply the new trial value by b, and if the result is less than a we subtract $\frac{1}{8}$, otherwise we add $\frac{1}{8}$. This process is repeated, adding or subtracting $\frac{1}{16}$, $\frac{1}{32}$, &c., until the quotient is determined to the required accuracy.

It is often said that the division of a by b can be performed by first calculating the reciprocal of b by means of the iterative formula $x_{n+1} = x_n(2 - bx_n)$ and then multiplying by a. The difficulty in using this method with a machine with a fixed decimal or binary point, however, is that $1/b$ is not necessarily within the range of the machine, even though b and a/b may be; in fact, if numbers in the store are all less than 1, the reciprocal of a number is necessarily out of range. This means that k/b, where k is a scale factor, must be handled inside the machine instead of $1/b$, and k must be chosen according to the magnitude of b. The necessary preliminary operations for determining the appropriate value of k add considerably to the length of the subroutine.

A method of calculating a/b (where a and b are positive) which avoids this difficulty is to calculate sequences of numbers a_n and b_n from the following formulae:

$$a_{n+1} = a_n(2 - b_n),$$
$$b_{n+1} = b_n(2 - b_n);$$

a and b are used respectively for the starting values of the two sequences. It will be seen that

$$a_{n+1}/b_{n+1} = a_n/b_n = \ldots = a/b$$

and also that b_{n+1} tends to unity as n increases. It follows that a_{n+1} tends to a/b. Since the data are introduced into this process only at the beginning, it is repetitive rather than truly iterative, and rounding-off errors may accumulate. It is not therefore to be recommended when the values of a and b are given exactly and high precision is required. If, however, a and b are rounded-off numbers, the method will give the quotient with little loss of accuracy.

Elementary Functions. Under this heading are included square roots and cube roots, trigonometrical and inverse trigonometrical functions, exponentials and logarithms, &c. Square roots and cube roots are best obtained by means of an iterative process based on the Newton–Raphson formula, and the other functions are best obtained from a power series. It does not usually pay to use methods which depend on interpolating, by means of conventional interpolation formulae, into a table placed in the store. Taylor series can be used for the power series, but frequently some economy, both in storage space and computing time, can be effected by using a Chebyshev series. Although methods making use of a power series are here differentiated from those making use of a table, it is legitimate to regard the coefficients of the series as a highly condensed table and the series itself as an interpolation formula.

Methods based on 'digit-by-digit' processes may also be used for subroutines in this category and, although slow, are sometimes useful when storage space is short.

Numerical Processes. Among the subroutines in this section may be one or more for interpolation, although, as already mentioned, this is not a common operation in a high-speed computer. Processes which make use of function values rather than differences are most suitable; Neville's process, in particular, has the advantage that it can easily be adjusted by means of a preset parameter to provide interpolation of any desired order.

Subroutines for numerical quadrature also come under the present heading. On the whole the more accurate formulae are to be recommended for use with an automatic computing machine, rather than those which depend on dividing the interval into a large number of

strips. In particular, it has been found that the Gauss formula, in which the strips are of unequal width, is particularly useful when the function to be integrated is defined by a formula and can be evaluated by means of an auxiliary subroutine. It is no more difficult to write a subroutine based on the Gauss formula than one based on the Newton–Cotes formula, but, since for the same accuracy half as many function values have to be computed, the subroutine is twice as fast and takes slightly less space in the store. The unequal spacing of the ordinates, which is a decided inconvenience in ordinary computing practice, is no disadvantage when using an automatic machine. An incidental advantage of the Gauss formula is that function values at the ends of the range of integration are not required; special provision need not, therefore, be made for obtaining those values when the formula given for the function takes on an indeterminate form at one of the ends. An example would be the evaluation of

$$\int_0^{\pi/2} \frac{\sin x}{x} \sqrt{(1+x^2)}\, dx.$$

A long range of integration can be covered either by the use of, say, a 12- or 14-point Gauss formula (these are equivalent to fitting polynomials of the 23rd and 27th degree respectively), or by the repeated application of, say, a 5-point formula.

It is convenient also to have in the library subroutines suitable for integrating a function known only at equal intervals of the argument. Such a function may have been obtained, for example, by solution of a differential equation, and the integral will probably be required as a function of its upper limit.

A library should certainly contain one or more subroutines for solving ordinary differential equations. Since any ordinary differential equation can be expressed as a set of simultaneous first order differential equations (not necessarily linear), a subroutine which will solve such a set of equations will, subject to sufficient storage capacity being available, enable any ordinary differential equation to be treated. It has been found that the Runge-Kutta process, or one of its variants, is particularly suited for use with an automatic computer. In this method, values of the function at the end of a given interval are computed from their values at the beginning, no use being made of values of the function at the beginnings of earlier intervals, even if these are available. This gives the method the important advantage that no special procedure is necessary either when starting or when the length of the interval is changed. A further advantage, important when a large number of simultaneous equations are being solved, is that very little storage space is required to hold function values. Since, however, the method does not take advantage of any information

which may be available about earlier values of the function, it is relatively slow compared with methods using backward or central differences, and there is a case for having these methods represented in the library as well. In certain cases, it may be convenient to start the integration with the Runge–Kutta method and to change over, after the first few steps, to a faster method. This procedure would be particularly well adapted to a machine in which subroutines can be brought rapidly into the high-speed store from an auxiliary store.

Matrices and linear equations. This section includes subroutines for the multiplication of matrices, the solution of sets of simultaneous linear algebraic equations, the inversion of matrices, and the determination of characteristic values and vectors. In addition to subroutines intended for inclusion in longer programmes, the library can usefully contain complete programmes for carrying out several of these processes. The solution of simultaneous equations can be performed either by direct methods, that is, by successive elimination of the variables, or by indirect methods, that is, by the application of some iterative or repetitive procedure. The same applies to the inversion of matrices. It would be out of place here to discuss the relative merits of direct and indirect methods, but it may be remarked that experience tends to show that, when an automatic computing machine is used, direct methods often have much to be said for them, even in cases where certain indirect methods would seem at first sight to have an advantage.

It is likely that the subroutines and programmes in this section of the library will exist in two forms. In each case one form will be suitable for use when the matrices are small enough for all their elements, together with the necessary orders, to be accommodated in the high-speed store at one and the same time. The other form will enable larger matrices to be handled by making use of the auxiliary store. In general, programmes and subroutines of the second form will be slower in operation than those of the first form; the amount by which they are slower will depend on the properties of the auxiliary store and the time taken to transfer blocks of numbers to and from it.

Interpretive subroutines. This section of the library may contain subroutines for facilitating the use of double-length numbers, complex numbers, and numbers expressed in floating-decimal or floating-binary form.

MULTI-ADDRESS CODES

The reader will remember that orders in the EDVAC code contain four addresses; in arithmetic orders, however, only three of these refer to operands, the fourth giving the address from which the next order is to be taken. Such a code will be referred to as a 3 + 1 address code. The extra address, giving the location of the next order, is of

value in machines using a delay store in that it enables *minimum access* or *optimum* coding to be undertaken; some remarks on this subject will be made in the next section.

Many machines use three-address codes, but take orders from the store consecutively in a similar manner to the EDSAC. Coding for such a machine will be illustrated by taking the same example as has been used above, namely the calculation and placing in storage location 0 of the scalar product of two 50-dimensional vectors whose components are in storage locations 100, 101, 102, . . ., 149, and 300, 301, 302, . . ., 349. The orders which will be used are as follows:

A *l m n* Add the number in storage location *l* to the number in storage location *m* and put the result in *n*.
V *l m n* Multiply the number in storage location *l* by the number in storage location *m* and put the result in *n*.
E *l m n* If the number in storage location *l* is equal to, or greater than, the number in storage location *m*, transfer control to storage location *n*; otherwise proceed serially.
F 0 0 *n* Transfer control to storage location *n*.

The programme is as follows:

	10	A	17	18	12	set variable order in 12
	11	A	18	18	0	clear 0
15→	12	(V	100	300	1)	add product to partial sum; variable order
	13	A	0	1	0	shown as on first repetition
	14	A	12	20	12	increase variable order
	15	E	19	12	12	test for end, if not repeat from 12
	16	F	0	0	21	jump to next part of programme
	17	V	100	300	1	pseudo-orders (numerical value of P is zero so that C(18) = 0)
	18	P	0	0	0	
	19	V	149	349	1	
	20	P	1	1	0	
	21		. . .			continuation of programme

This example shows how little difference there is in principle between coding in a three-address code and coding in a single-address code. A good deal of variety is possible in three-address codes, but it is not necessary here to give further examples. Two-address codes, in which use is made of an accumulator, have also been used. No one who has had experience with programming for a machine with a single-address code is likely to find himself long at sea when he encounters a machine with a multi-address code. Furthermore, the various aids to programming discussed in this chapter and in Chapter 7 can all be applied to multi-address codes.

It will be observed that the above programme contains about half as many orders as the corresponding programme written in a single-address code given on p. 89. Since, however, three-address orders are about twice as long as single-address orders, the total storage space occupied is much the same. A three-address code is particularly efficient for adding two numbers together and placing the result in the store, since this needs only one order compared with three

orders in a single-address code. However, if four or more numbers have to be added together it will be found, when allowance is made for the fact that single-address orders are individually half as long as three-address orders, that the single-address code is more efficient. This is illustrated by the following example:

```
A 100          A 100   101   0
A 101          A 102     0   0
A 102          A 103     0   0
A 103
T   0
```

In long programmes a three-address code usually shows a small advantage as regards storage space occupied; this is because operations involving three operands only are common.

Minimum-access Coding

If a machine which uses an ultrasonic delay unit or a magnetic drum (see Chapter 5) for the main store is designed so as to take its orders from consecutive storage locations, a relatively large amount of time is consumed in waiting for orders and numbers to emerge from the store. Such machines are, therefore, often designed so that the programmer can specify in each order the location from which the succeeding order is to be taken. It is then possible to arrange the programme so that orders—or at any rate a fair proportion of them—are so located in the store that they emerge with only a short delay when they are called for. By the exercise of some ingenuity, it is further possible to arrange that constants and certain other numbers used in the calculation are also located so that they can be obtained, when required, with a minimum of delay. Relatively little can be done, however, to minimize access time in the case of numbers which are required more than once during the course of a calculation, or which appear as intermediate results, unless a number of special registers (for example, short mercury tanks or static magnetic delay lines), which permit of rapid access, are provided. At least four or five such registers are needed to make effective minimum-access coding possible, and some ten or twenty are really desirable; even so, minimum-access coding for some problems will be far from straightforward, and a good deal of trial and error will be needed. Minimum-access coding can, in cases where it is applicable, enable very high computing speeds to be obtained, but not without much labour on the part of the programmer.

In order to give an example of minimum-access coding I shall consider a hypothetical computer working in the serial mode and fitted with a magnetic drum as its main store. One track on the drum will be sufficient to accommodate the programme considered, and it will be assumed that this track contains 128 storage locations. The order

code will be the same as that given on pp. 87–8, with the addition of an extra address to specify the location from which the next order is taken; for example:

A *n m* Add the number in storage location *n* into the accumulator and take the next order from storage location *m*.

It will be assumed that orders are executed in the word time following that in which they are read from the drum, and that the next order can, if required, be read immediately afterwards; thus if the order is in storage location *p*, then the fastest operation is obtained if $n = p + 1$ and $m = p + 2$. Multiplication is an exception to the foregoing rule and takes 32 word times to complete, so that for fastest operation *m* must be equal to $n + 32$; if $m = n + 31$, a whole revolution of the drum is wasted.

The example which will be taken is to calculate $a_1b_1 + a_2b_2 + a_3b_3$ and $a_1^2 + a_2^2 + a_3^2$ where the six numbers a_1, a_2, a_3 and b_1, b_2, b_3 are given. A routine which performs the calculation in just under three drum revolutions is given below; this routine has been drawn up on the assumption that no registers to which rapid access may be obtained are available. The given numbers are assumed to be initially in storage locations 1, 3, 42, 44, 82, and 84; these locations were chosen with a view to minimizing access time.

Storage location	Content			Explanation
	Function	Address of operand	Address of next order	
0	H	1	2	executed in 1st rotation
1	a_1			initial content
2	V	3	41	executed in 1st rotation
3	b_1			initial content
4	T	6	40	executed in 2nd rotation
5	V	6	39	,, ,, 3rd ,,
6	(a_1)			planted ,, 2nd ,,
.				
.				
.				
39	H	42	46	executed in 3rd rotation
40	A	42	45	,, ,, 2nd ,,
41	H	42	43	,, ,, 1st ,,
42	a_2			initial content
43	V	44	81	executed in 1st rotation
44	b_2			initial content
45	T	47	80	executed in 2nd rotation
46	V	47	79	,, ,, 3rd ,,
47	(a_2)			planted ,, 2nd ,,
.				
.				
.				
79	H	82	86	executed in 3rd rotation
80	A	82	85	,, ,, 2nd ,,
81	H	82	83	,, ,, 1st ,,
82	a_3			initial content

Storage location	Content: Function	Content: Address of operand	Content: Address of next order	Explanation
83	V	84	119	executed in 1st rotation
84	b_3			initial content
85	T	87	88	executed in 2nd rotation
86	V	87	122	,, ,, 3rd ,,
87	(a_3)			planted ,, 2nd ,,
88	H	1	5	executed ,, 2nd ,,
.				
.				
.				
119	T	120	121	executed in 1st rotation
120		$(a_1b_1+a_2b_2+a_3b_3)$		planted ,, 1st ,,
121	A	1	4	executed ,, 1st ,,
122	T	123	—	,, ,, 3rd ,, (last order of sequence)
123		$(a_1^2+a_2^2+a_3^2)$		planted in 3rd rotation

This routine contains 20 orders and would take 59 msec to operate if the drum rotates at 3,000 r.p.m. This is an average of 3 msec per order. If the orders had been placed in consecutive storage locations each would have taken about 20 msec. The gain in speed is thus quite considerable.

The example just considered is a favourable one for showing what can be done even if no registers with rapid access are available. If five such registers were available it would be possible to complete the calculation in about $1\frac{1}{2}$ revolutions, very little time being lost in waiting for numbers or orders to emerge from the store.

Only those parts of a programme which account for the greater part of the computing time need be coded for minimum access. Careful attention to this point will reduce the heavy labour of minimum-access coding to a minimum. It is worth paying special attention to the coding of library subroutines in order to make them as fast as possible.

In minimum-access coding the use of short loops of orders is discouraged, since the execution of each loop takes at least one revolution. There is thus a tendency for cycles of orders to be written out several times. When the same constant is required at several points in a programme, it will usually be necessary to put it in more than one place in the store. For these reasons, programming for minimum access time tends to require more storage space than ordinary programming. This point should be borne in mind when comparing the number of storage locations available in different machines, some of which are designed for minimum-access coding and some not. Again, when minimum-access coding is used the orders themselves are longer than they otherwise would be, on account of the extra address required to specify the location of the next order. In a machine with a

single-address code this may well mean that only one order, instead of two, can be accommodated in the space taken by a single number. The subject of minimum-access coding will be taken up again and discussed in relation to machine design in Chapter 7.

THE USE OF AN AUXILIARY STORE

When programming for a machine which has no auxiliary store, a programmer usually finds himself under the necessity of compressing the programme into as small a number of orders as possible, so that it may be accommodated in the high-speed store. Many problems have to be dealt with in a series of separate stages. Each stage is performed by means of a separate programme, and numerical results obtained in one stage are put back into the machine in another. Problems in which a large amount of numerical information has to be handled—for example, those involving the manipulation of large matrices—are frequently very difficult to deal with in such a machine, and the importance of an auxiliary store in connexion with such problems will be apparent. Apart, however, from making it possible to handle large quantities of numbers, the presence of an auxiliary store enables longer and more complicated programmes to be used than would otherwise be possible. It is with this aspect of the use of an auxiliary store that this section is concerned.

The use made of an auxiliary store depends very much on the speed with which blocks of numbers can be transferred from it to the high-speed store, and vice versa. A machine in which such transfers are relatively slow—for example, one using magnetic tape or wire for auxiliary storage—will probably have a large high-speed store, and the programmer will divide his programme into stages in much the same way as he would do if no auxiliary storage were available. The programmes for the various stages will, however, be called down from the auxiliary store when required, so that the various stages will follow one another automatically with such repetitions as are necessary. Numbers computed in one stage and required in another will either be placed temporarily in the auxiliary store or, if there is room, held in the high-speed store. As an example, consider a programme concerned with the calculation of the resonant modes of oscillation of an aircraft wing. The first stage is to take into the machine a set of numbers specifying the size, shape, and elastic constants of the wing. From these the machine calculates the elements of a matrix whose latent roots give the required periods. In the second stage these latent roots are determined. The programmes for the two stages are held in the auxiliary store and transferred in turn to the high-speed store. Possibly the determination of the latent roots would be only one step in a calculation of greater complexity. In this case further stages would follow the two just described.

In the example just given the programmes for the two stages are completely independent, and may be constructed along the lines indicated earlier in this chapter. Each programme has embodied in it such library subroutines as are required, and the same library subroutine may figure in one or more programmes. Sometimes, instead of transferring an entire new programme from the auxiliary store when a new stage of the problem is to be started, it is sufficient to modify the programme already in the high-speed store by overwriting part of it with a new sequence of orders from the auxiliary store. It is obviously desirable, when an auxiliary store with a relatively long access time is used, that orders should be transferred from the auxiliary store to the high-speed store in blocks which are sufficiently large to keep the machine busy for an appreciable period—if possible of the order of minutes. If this can be done the total time spent in securing access to the auxiliary store is a small fraction of the total computing time.

In the UNIVAC it is possible to transfer a block of orders from the auxiliary store (which consists of a group of magnetic tapes) to a set of buffer registers while the rest of the machine is working normally. The buffer registers can then be discharged into the main store at high speed. In suitable problems, the work may be so planned that computation is halted only during the short periods when this transfer is taking place.

If the machine has an auxiliary store with a short access time—for example, a magnetic drum—it becomes economic to call down individual subroutines from the auxiliary store each time they are required. What has been said earlier in this chapter about the construction of subroutines will largely apply to programming for such a machine, but the problems of programme assembly are quite different. The high-speed store (which may be relatively small in capacity) assumes the role of a temporary repository for the section of the programme being executed at a particular moment (together with the numbers being operated on), the programme as a whole being contained in the auxiliary store. This style of programming will be illustrated by describing one of the procedures which has been used in the Computing Machine Laboratory of Manchester University. The machine installed there has an auxiliary store consisting of a magnetic drum with 256 tracks, each capable of holding 128 orders. The contents of one track can be transferred to the high-speed store in 35 msec.

In this system the programme is divided into a number of *routines*, each of which occupies not more than one track on the magnetic drum. Before being operated, each routine is transferred to a part of the high-speed store set aside for that purpose, overwriting the routine which was there before. Each routine which stands in the

high-speed store at any stage of the programme must contain the orders necessary to replace itself by another routine. In calculations which can be conveniently coded on one level—that is, without making use of closed subroutines or their equivalent—this can be very simply done by placing at the end of each routine a *magnetic instruction* which will call down the next routine. In a more complicated calculation which involves the use of subroutines, it is desirable to make use of a *routine changing sequence*. This is a sequence of orders held permanently in the high-speed store. When a point is reached in the routine currently being operated at which a closed subroutine must be called in, control is sent first to the routine changing sequence, a record of the point reached in the current routine being first made in one of the B-registers. The routine changing sequence then causes the subroutine to be transferred from the auxiliary store to the high-speed store and control to be sent to it at the correct entry point. Before this happens, however, the routine changing sequence constructs, and plants within itself, a link by means of which the original routine can be brought back into the high-speed store and control sent to it at the place immediately following that previously reached. Thus, when the closed subroutine has done its work, control passes back to the routine changing sequence, which calls down the original routine and returns control to it.

ERROR DIAGNOSIS

Reference has already been made to the fact that a programme when first written is likely to contain one or more 'blunders'. Some of these can be found by careful checking before the programme is punched, especially if this checking is done by another person. However, blunders will frequently remain undetected until the programme is tried. It is desirable that this fact should be recognized and that suitable means for error diagnosis should be provided. They may either be built into the programme from the start, or provision may be made for bringing them into action without loss of time if the programme fails.

When high-speed automatic computing machines were first about to be put into action, most people thought that a satisfactory way of finding blunders would be to make the machine go order by order under the control of a push-button, and to inspect the contents of the store or arithmetic registers at frequent intervals. It was soon found, however, that this procedure consumed a large amount of machine time and that frequently the machine was standing idle while the programmer tried to work out what, if anything, had gone wrong during the last step. Moreover, even if means were provided for stopping the machine somewhere near the point at which the error was thought to occur, much pushing of the button was necessary in order to reach

the critical point in the programme. It is now generally agreed that a more satisfactory method of error diagnosis is to make the machine repeat the calculation and at the same time print out more information than the programme is designed normally to call for. The extra printing can take place either during the course of the calculation or when the programme has stopped. The information can then be studied at leisure by the programmer while the machine is proceeding with other calculations.

Blunders are of two kinds. There are those which affect seriously the structure of the programme and cause the machine to behave in a manner which is totally unexpected—perhaps to print meaningless symbols or to stop altogether—and there are those which result in the machine appearing to perform the calculation in the way intended but giving results which, when compared with check results calculated independently, are found to be in error by a small or large amount. An example of the former class is the insertion of an incorrect address in an order transferring control, with the result that the correct sequence of operations is broken. A blunder of the second class might cause the capacity of the accumulator to be exceeded, with the result that one or more digits were lost off the left-hand end; alternatively, the number in the accumulator might become small, with a resulting loss of accuracy upon rounding-off. A simple method which is often effective in finding blunders, especially those of the first class, is to print out the contents of the store when the machine has been stopped by hand or has stopped of its own accord. This is sometimes known as the *post-mortem* method. A refinement, which saves printing time, is to make the machine print out only those numbers and orders which have been changed during the programme. When the machine stops of its own accord a note should be made of the storage location from which the last order was taken, although it is often found that this information is of little help in tracing the blunder, since, once off the right track, the machine may perform a large number of orders before finally stopping.

In the EDSAC code there are some function letters which do not correspond to machine operations and, if the digits corresponding to one of these letters enter the control circuits of the machine as a result of control being sent to a location which contains a number instead of an order, the machine stops. Some other machines in similar circumstances perform no operation but pass to the next location instead of stopping. Although it was not foreseen when the EDSAC was designed, this particular property has proved a great help in error diagnosis, since, if there is something seriously wrong with the structure of a programme, it will not be long before control is sent to a location containing a number, and there is then a good chance that the machine will stop. It is suggested that this feature should be in-

corporated in the design of future machines. If the function digits are the most significant digits of an order (and this is a good reason for choosing them to be), it is convenient if the machine stops when they are all zero, since a large proportion of the numbers in the store are usually small or medium in size and have their most significant digits as zero. If complements are used for negative numbers, it is similarly convenient if the machine stops when the function digits are all 1s. In the event of control being sent, as a result of a blunder, to a part of the store which contains numbers, the machine is then unlikely to run on very long before stopping.

Subroutines of the interpretive type can be applied to error diagnosis. The error-diagnosis subroutine is placed in the store along with the programme. The orders of the programme do not themselves enter the control circuits of the machine in the normal way. Instead, control is sent to the subroutine, which treats the orders of the programme as interpretive orders in the manner described on p. 103. They are extracted one by one and caused to be performed in the sequence in which they would be performed if the programme were being executed normally. At the same time the error-diagnosis subroutine causes extra information to be printed. For example, the content of the accumulator may be printed every time the accumulator is about to be cleared; this would help in locating a blunder of the second class. Alternatively, the function letter of each order may be printed as that order is executed; this will be found to give a surprising amount of information about the structure of the programme and to form a powerful method of locating the point at which the machine first failed to do what was expected on account of a blunder of the first class.

Interpretive methods of error diagnosis are most useful in connexion with short programmes. If they are to be used with longer programmes some means must be provided of preventing the interpretive subroutine from printing a great deal of unnecessary information about sections of the programme in which no error is suspected. For example, it can be arranged that no printing takes place when orders from certain parts of the store are being executed, or when closed subroutines are being operated; alternatively, a 'delayed start' can be arranged so that printing does not begin until a specified point in the programme is reached. Complications of this kind, however, make the error-diagnosis subroutine more troublesome to use and, if they are carried too far, get out of hand. Experience has shown that those methods of error diagnosis which are most straightforward to use are the most successful in practice. Unfortunately, these do not always involve the least consumption of machine time. Much work remains to be done in developing improved methods of error diagnosis, and it seems probable that the programmer will need some help from the machine designer by way of the provision of special facilities.

THE INCORPORATION OF NUMERICAL CHECKS IN PROGRAMMES

Since machines are always liable to make errors—more especially errors of a random kind—it is necessary, when planning a production schedule for a particular job, to make sure that an adequate system of numerical checking is included. How frequently the checks should be made depends on circumstances. In general, it is unwise to let a machine run for more than 15 or 30 minutes without the application of some numerical check, if the occurrence of an error at any time during that period can render the entire work useless. However, it is not worth while putting in frequent checks if more time is consumed in performing them than would otherwise be lost in re-running. Naturally, much depends on the liability of the machine to make sporadic errors, and this may vary from time to time. Since the object is to avoid waste of time, the decision on how frequently to incorporate checks does not depend very much on the speed of the machine, provided that the rate of occurrence of sporadic errors is fixed, and checks are likely to be wanted at much the same intervals in the case of a fast machine as in the case of a slow one. Since, however, the fast machine will do much more computing between checks, the nature of the checks applied is likely to differ.

The use of very frequent numerical checks in the programme makes it possible to do useful work with a machine which would otherwise be much too unreliable to use. It is necessary, however, to be able to arrange the programme so that, if a check fails, the relevant part of the calculation is automatically repeated. This means that the machine must have a certain amount of reliable storage, so that it can go back to an earlier stage in the calculation. An example of a machine with which this kind of operation would be possible is an electronic machine with a reliable auxiliary store (for example a magnetic drum) and a less reliable high-speed store. Another example is a relay machine which uses paper tape both for storing the programme and for intermediate storage of numbers.

CONVERSION ROUTINES

So far this chapter has dealt with established systems of programming and techniques which are, in some form or other, in general use. Something will now be said about more elaborate methods, intended to simplify the construction of programmes by making the machine itself do part of the work. Many of these methods are at present the subject of experiment, and all that can be done here is to give some indication of the lines along which the experiments are proceeding. There is not yet general agreement among users of computing machines as to the value of the proposed systems, but an attempt at a provisional evaluation will be made in Chapter 7.

The principle on which all the methods discussed here are based is that the programmer should be enabled to write his programme in a form which is convenient for himself, and that it should then be converted automatically by the machine into a correctly assembled sequence of regular machine orders. The conversion is performed by the machine, acting under the control of a *conversion routine*. The converted programme is either placed in the store for immediate execution or is passed temporarily out of the machine for subsequent reinput when it is required to be run. The initial input routine described earlier in this chapter may be regarded as a simple conversion routine, since it enables the programmer to use relative systems of numbering for addresses, although these are not provided for by the order code of the machine. The conversion routines with which I am here concerned, however, are of much broader scope.

A conversion routine may help a programmer in two ways. It may provide him with new orders to supplement or replace those contained in the order code of the machine, and it may relieve him of the task of assembling the components of a programme—master routine, subroutines, and number sequences—and of inserting the necessary cross references, for example, the addresses in the orders which send control from the master routine to a subroutine.

An example of an order, supplementary to the order code of the machine, which might be provided is one for replacing the content of the accumulator by its modulus (it is assumed that the basic order code of the machine does not contain such an order). This would be written on the programme sheet and prepared for input to the machine—for example, by punching on paper tape—in a form very similar to that of any other order, but it would be recognized by the conversion routine, which would replace it by the sequence of four or five orders required to take the modulus. Another example is an order for multiplying the content of the accumulator by some specified number x, again assuming that no basic operation of this kind is provided. Here, in addition to inserting a sequence of orders in the programme, the conversion routine would place the number x in the next available location of a sequence of storage locations allocated for the purpose, and would insert the address of this location into the order which, when the programme was operated, would perform the multiplication. A conversion routine could also be used to facilitate the handling of complex quantities. The programme would be written as though the quantities concerned were real, and the conversion routine would replace each arithmetic order by the short sequence of orders necessary to perform the required operation on the real and imaginary parts of the numbers concerned. It will be seen that, in the latter example, the end achieved is, from the point of view of a programmer, exactly the same as if an interpretive routine were used.

There is the difference, however, that the expansion of the orders as written into sequences of machine orders takes place once for all, when a conversion routine is used, and every time the section of the programme concerned is operated, when an interpretive routine is used. Thus the converted programme is lengthy, but, on the other hand, its operation is fast, since no time is taken for interpretation. Furthermore, a conversion routine, unlike an interpretive routine, does not occupy space in the store during the running of the programme. Although I have drawn a clear distinction here between conversion and interpretation, many of the more elaborate schemes which have been used, or proposed for use, with a view to simplifying programming, contain features of both types.

An early example of the use of a machine to perform the purely clerical tasks involved in the assembly of a programme, was an *assembly subroutine* which was constructed for the EDSAC in 1950; this was capable of assembling a master routine and a collection of subroutines into a complete programme. The assembly subroutine was punched first on the input tape, so that it went into the store in advance of the programme; it was followed by the master routine and the various subroutines in the order in which they were to go into the store. At the head of each was punched a special control combination. The orders from the tape were read in the ordinary way and placed in consecutive locations. When each control combination was encountered reading of orders was temporarily halted and a record made of the location in which the next order (being the first order of the master routine, or of a subroutine) was to go. The record consisted of an order which would transfer control to the first order of the master routine or subroutine, and the sequence of such orders (which were placed in a sequence of consecutive storage locations set aside for the purpose) constituted a directory. Control could be caused to go to the master routine, or to one of the subroutines, by sending it first to the appropriate order in the directory. The code letter ϕ was used to define a relative numbering system for the orders in the directory, so that the third subroutine (for example) punched on the tape would be called in by the pair of orders

m	A m F
$m+1$	F 3 ϕ

The address in the F order depended only on the number allocated to the subroutine and not on the place it occupied in the store. It could, therefore, be written in its final form right from the beginning and did not need to be altered if changes were made to the programme which necessitated the repositioning of the subroutine.

Floating Addresses

It has already been pointed out that by using a system of relative numbering the programmer gains a substantial advantage in that he is enabled to draw up subroutines and sequences of numbers without taking into consideration the places in the store which these components will occupy when finally assembled to make a complete programme. The use of *floating addresses* enables him to be set free to an even greater extent from the limitations of the basic numbering system of the machine.*

If a programme is examined it will usually be found that only a small fraction of the orders contained in it are referred to in other parts of the programme. A system whereby every order is numbered is, therefore, unnecessarily rigid, and carries with it the disadvantage that the programmer must be prepared to undertake extensive renumbering whenever extra orders are inserted into the middle of a programme, either to modify its action or to correct an error. The use of floating addresses makes it unnecessary for the programmer to give serial numbers to orders which are not referred to at other points in the programme; only those orders which are referred to are allocated serial numbers, and these need not necessarily occur in numerical order. The programmer is then completely relieved of the necessity of renumbering when he makes alterations, and he can write the addresses specified in the orders in their correct form from the beginning. He is not impelled, as he would otherwise be, to postpone the insertion of the addresses until he is quite sure that his programme is in its final form. The replacement of the floating addresses by absolute addresses is done by a conversion routine after the programme has been taken into the machine and before it is operated.

A programme for forming and printing

$$\sum_{0}^{99} \alpha_n^2,$$

where $\alpha_n = \mathrm{C}(200 + n)$, might be written on the programme sheet in the form given below. It is assumed that the sequence of numbers α_n have been placed in advance in the store. The notation used for orders is similar to that of the EDSAC code, except that absolute addresses are written as numbers without any terminating code letter, and floating addresses are written as letters with subscripts.

* This section is concerned with machines in which orders are taken consecutively from the store for execution. The same considerations do not apply when programming for a machine with a 1+1 address or a 3+1 address order code.

Location in store	Order		
	T	0	clear 0
a_3	S	b_2	part of counting sequence
	T	b_3	
a_1	A	0	accumulate Σa_n^2 in 0
a_2	H	200	
	V	200	
	T	0	
	A	a_1	increase address of H order
	A	b_1	
	T	a_1	
	A	a_2	increase address of V order
	A	b_1	
	T	a_2	
	A	b_3	part of counting sequence
	A	b_1	
	G	a_3	
a_4	A	a_4	call in closed print subroutine with first order in c_1
	F	c_1	
	Z		stop
b_1	P	1	constants
b_2	P	100	
b_3	(	)	counting number

The programme could, for convenience, be punched on to paper tape in exactly the form in which it is written, with symbols for spaces, letter and figure shift, and lay-out included. A fair copy of the programme could then be obtained, for checking and record purposes, by passing the tape through a tape reader connected to a printer. Similar arrangements would be possible with magnetic tape or punched cards. If no inferior figures were available the suffixes could be punched as ordinary figures following the letters.

Since the task of replacing the floating addresses by absolute addresses involves nothing more than the application of a series of rules, there is no serious difficulty in drawing up a programme to do it. The floating addresses cannot, however, be dealt with in sequence, as they are encountered when the programme is being read, since floating addresses used in some of the earlier orders may be those of later orders in the programme. The correct values to be assigned to them will not, therefore, become known until later. Also, unless wholesale wastage of storage space is permitted, the allocation of values to floating addresses in the various letter sequences, *a*, *b*, *c*, &c., cannot be made until it is known how many numbers there are in each sequence. The system adopted, therefore, is for the programme to be read as it stands into the auxiliary store. It is then examined by the conversion routine, which runs through the orders as many times as may be necessary, and builds up a directory giving the values of the various floating addresses. Finally, these values are incorporated in the pro-

gramme by the conversion routine, and the programme is placed in its ultimate location in the high-speed store.

It will be observed that, in addition to facilitating the task of the programmer by providing him with a flexible system of numbering for orders, the use of floating addresses also solves the problem of assembling the master routine and the various subroutines into a complete programme, and renders the use of a special assembly subroutine unnecessary. All that is required is that the first order of the master routine and the first orders of the subroutines shall be numbered with floating addresses. In the example just given, the closed subroutine used for printing was treated in this way.

Instead of regarding the symbols a_1, a_2, &c., and b_1, b_2, &c., as standing for the addresses of locations in the store, one may regard them as standing for the quantities stored in those locations. The order A a_1 with this interpretation means 'add a_1 into the accumulator'. A symbol such as a_1 written against a number or an order is regarded as a label identifying that number or order as the quantity a_1. If the programme given above is examined with this interpretation in mind, it will be found that no ambiguity of any kind arises; in fact, the uninitiated would probably regard it as a more natural interpretation than the original one.

The above remark may be put more formally by saying that, since there is a (1, 1) correspondence between the storage location and the number or order contained in it, any label given to a location can equally well be applied to the content of that location. However, when, as in ordinary programming, the storage location is referred to by its serial number, the fact that the content of the location may also be a number makes it necessary to preserve a distinction between the two. This difficulty is avoided with floating addresses by the convention that letters with suffixes are used instead of serial numbers to identify locations.* The advantage of using the labels to refer to the contents of locations rather than to the locations themselves becomes specially apparent when floating addresses are used. The labels attached by the programmer to numbers and orders are not then subject to change whenever the programme is revised, as they are when less flexible systems of numbering are used.

The Incorporation of Open Subroutines

An advantage claimed earlier for the use of closed subroutines was that they enabled the master routine to take on a very simple form, the closed subroutines being represented in it by nothing more than the pairs of orders which called them in. Once it has been decided to use a conversion routine, however, this particular advantage may be

* Letters were used to refer to storage locations in the Bell relay machines and letters with suffixes in the Harvard Computers Mark III and IV.

obtained in another way. The subroutines themselves can be of the open variety and can be held in a section of the auxiliary store reserved for the purpose. At the point in the master routine at which the programmer wishes a particular subroutine to be incorporated, he writes down suitable symbols specifying the subroutine in question. These are interpreted by the conversion routine which extracts the appropriate subroutine from the auxiliary store and incorporates it in the programme. Thus, in the example given above, instead of writing

$$\begin{array}{ll} a_4 & \mathrm{A}\ a_4 \\ & \mathrm{F}\ c_1 \end{array}$$

he writes (say)

SR P4

indicating that the subroutine bearing the reference number P4 is to be incorporated at that point. He then writes the remaining orders of the master routine. If this method is used, the written form of the master routine is just as simple as if closed subroutines are used. When it reaches its final position in the high-speed store, however, it has been expanded out of all recognition by having open subroutines embodied in it. An advantage of this procedure is that the orders for calling in the closed subroutine and the orders, within the subroutine, for forming and planting the link are not required; this results in a slight saving of computation time and avoids unnecessary use of storage space. It is in a way more logical that a programme, which has been written in a certain form to suit the convenience of the programmer, should be converted on input to a form which is simpler from the point of view of the machine, rather than that it should retain the same form inside the machine.

The situation is somewhat different when a subroutine is required at more than one point in the master routine. Here the use of closed subroutines is as efficient as any other method that may be devised for switching control to the subroutine from various points in the programme and for causing control to return. If subroutines were stored in the auxiliary store in open form, therefore, it would be desirable to provide some means whereby they could be converted to closed form on being placed in the high-speed store. The programmer would then head his master routine with code groups specifying which subroutines were to be treated in this way; it would even be possible to design the conversion routine so that it would detect which subroutines were used more than once and automatically convert them to closed form.

In the remarks which have been made above about the use of conversion routines, it has been assumed tacitly that the whole programme can be accommodated in the high-speed store, or that a preliminary division into sections, each of which can be so accommo-

dated, has been made by the programmer. The design of a conversion routine which would itself perform this division, and introduce into the converted programme orders for bringing the sections into the high-speed store as they are required, would not be difficult if the speed of operation of the final programme were of no account. The securing of a reasonable degree of efficiency as regards speed of operation, however, presents grave difficulties, and, although some work has been done on the subject, notably at M.I.T., much more remains to be done before it can be said that a practical system is in sight.

THE CHOICE OF A SYSTEM OF PROGRAMMING

It is very desirable that all the programmers in a group should make use of the same, or substantially the same, methods. Not only does this facilitate communication and co-operation between the members of the group, but it also enables their individual experience more readily to be absorbed into the accumulated experience of the group as a whole. The best way of ensuring that programmers use a preferred system of programming is to make it easier for them to do that rather than anything else. When a programming group is being built up for a new machine, therefore, careful thought should be given to the design of the system which is to be used, and the group should be organized to produce, on a common plan, the input routines, basic library subroutines, and error-diagnosis subroutines. Provided that these offer satisfactory facilities, it will be much easier, once they are prepared, for an individual programmer to make use of them rather than to set about designing a system of his own.

It will first be necessary to secure agreement on certain conventions —particularly as to the method of calling in subroutines and the specification of parameters and arguments. There should, however, be as few of these conventions as possible. Elaborate schemes, however well they may appear, at the time they are being drawn up, to cater for all contingencies, are likely to prove irksome and restrictive in practice. During the initial stages alterations to the original scheme will no doubt be necessary, but the temptation continually to modify and improve should be resisted. Accurate and confident programming can only be achieved within the framework of a stable system.

In addition to saving the programmer trouble by providing him with ready-made subroutines, the library serves the equally important purpose of providing a permanent storehouse of programming experience. Programmers who have evolved methods for dealing with particular situations should be encouraged to make the further additional effort of expressing the results of their work in the form of a group of subroutines for the library. Even if these subroutines are never used, as they stand, by future programmers, a detailed study of

the way they are constructed may be of marked assistance to programmers faced with similar but different problems. Nothing should be allowed to go into the library which has not been thoroughly tested, and which is not the result of practical experience as well as of deep thought, because it is this practical aspect of a library which gives it its peculiar value. It requires a certain conscious effort on the part of the programming group to ensure that the library is continually enriched, but I know of no other way in which experience may be conserved in a group which is subject, as most groups of programmers are, to changes in personnel.

CHAPTER 4

RELAY COMPUTERS

THIS chapter is concerned with automatic calculating machines in which telephone relays are used as switching and storage elements. Since a single relay can carry six or more pairs of change-over contacts, quite elaborate switching operations can be performed without making use of very large numbers of relays. Thus relays lend themselves very well to the construction of the control and arithmetic sections of large automatic calculating machines. Unfortunately they are subject to sporadic and transient faults arising from partial oxidation of the contact points or from the intrusion of particles of dust, and they are, by electronic standards, slow in operation. Moreover, when storage as well as switching requirements are taken into account it is found that the number of relays required is several times the number of valves in an electronic machine of similar capacity; thus a large relay machine is at least as costly as a comparable electronic machine and has the disadvantage of being very much slower in operation. Unless spectacular development in relay design takes place, therefore, very few relay computers are likely to be built in the future. Nevertheless, relay computing circuits are of some interest in their own right, and their study throws light on the general principles of computing machine design. Moreover, relays are still used in connexion with the input and output channels of computers, and in the auxiliary equipment used to handle punched paper tape and punched cards. Some discussion of relay circuits will therefore be given in this chapter. Special attention will be paid to the computers built by the Bell Telephone Laboratories on the one hand and the Computation Laboratory of Harvard University on the other. It is particularly interesting to compare these machines, since the design principles on which they are based are quite different.

Development in the field of relay computing devices began at the Bell Telephone Laboratories in 1938. A computer for handling complex numbers was demonstrated in 1940; this had no automatic control features, and was to be compared with a desk calculating machine. Later a relay interpolator, containing about 500 relays, and automatically controlled by means of a programme tape, was constructed. This machine had storage capacity for five five-decimal numbers. Shortly afterwards another small machine of somewhat more general application, and known as the 'Ballistic Computer', was built. This also was controlled by a punched tape and had storage capacity for 10 numbers; it contained about 1,300 relays. The

success of these small machines led to the construction of two large computing installations for American Government agencies. These installations are now described as Model V. Each consists of two machines which can, when necessary, be coupled together. The last machine of the series to be built was one installed at the Bell Telephone Laboratories, Murray Hill, N.J., and known as Model VI.

The Computation Laboratory of Harvard University have built one relay machine which was completed in 1948. This machine is now in the U.S. Naval Proving Ground at Dahlgren, Va., and is sometimes referred to as the Harvard Mark II machine.

One important difference between these two machines arises on account of the different attitudes taken by their designers to the inclusion of checking circuits. The Harvard Mark II machine follows in the tradition of the Automatic Sequence Controlled Calculator (Mark I) in that the whole onus of checking, including checks on the operations performed by the machine, is placed on the programmer. The designers of the Bell machines, however, did all they could to make them self-checking, and these machines will stop rather than make an error unless the circumstances are very much against them.

It was pointed out in Chapter 2 that, in general, the provision of checking devices increases the bulk and complexity of a machine, and hence tends to reduce its overall reliability. This objection carries less weight in the case of a relay machine than it does in the case of an electronic machine, because of the large number of contacts which may be fitted to a single relay, and the complicated circuit arrangements which this makes possible. Moreover, except in so far as their effect is to increase the number of relays required, checking circuits can be added to a relay machine without reducing the operating margins. On the other hand, in an electronic machine undesirable effects due to capacitative loading, and the imperfect switching action provided by all known forms of electronic gate, become more troublesome as the complexity of the circuits is increased.

Relays are especially prone to faults of a transient nature such as may be caused by a particle of dust becoming lodged between a pair of closed contacts and falling off again when the contacts open. For a relay machine to be useful, therefore, frequent checks must be applied either by 'duplicate' programming, in which every step of the calculation is checked by repetition with the operations performed in a different order, or by means of built-in checking circuits. In the former case the programme must be so designed that, when an alarm occurs, it is possible to go back and repeat the section of the programme which has failed. In the latter case, if the built-in checking circuits are sufficiently comprehensive, it may only be necessary to repeat one step. Usually, when there has been a stoppage, the machine will go successfully through the check point the next time. If it

does not, then the fact that the machine has been stopped just after making a mistake greatly eases the problem of locating the faulty relay. Indeed, without some assistance of this kind it would be extremely difficult to find faults in a machine with a large number of elements subject to intermittent failure.

It should be noted that if 'duplicate' programming is necessary all programmes are twice as long as they otherwise would be and take twice as much time to run on the machine. In these circumstances the inclusion of built-in checking circuits doubles the effective speed of the machine.

The above considerations show that quite a strong case can be made out in favour of building checking circuits into a relay machine. The case depends partly on the frequency with which intermittent or transient faults occur in such machines, and partly on the comparative facility with which checking circuits can be designed. Electronic machines differ from relay machines in both these respects, and there need be no inconsistency if a different attitude to checking circuits is taken in the two cases.

THE BELL COMPUTERS

The automatic computing machines constructed by the Bell Telephone Laboratories are all decimal machines. They use a coding system in which each decimal digit is represented in the scale of 5. Two digits in this scale are needed to represent a decimal integer between 0 and 9, and, since the more significant of these digits can take the values 0 or 1 only, the system of representation is described as *biquinary*. Each decimal digit is stored on a group of seven relays, of which two are used to store the first (binary) digit, on a one-out-of-two system, and the remaining five to store the second (quinary) digit on a one-out-of-five system. Another way of describing the system of representation used is to say that each decimal digit is represented by a seven-digit binary number according to the following code:

0	0100001
1	0100010
2	0100100
3	0101000
4	0110000
5	1000001
6	1000010
7	1000100
8	1001000
9	1010000

This form of representation has three advantages which caused it to be preferred to the excess-three code which had been proposed earlier in the Bell Telephone Laboratories. First, a good check that no error has occurred in the transfer of a number to a register can be obtained

by providing checking circuits on the register which give an alarm unless one and only one relay in each group of two or five is energized. Secondly, conversion of numbers to and from the decimal system is easily carried out. Thirdly, a relay adding circuit, working on the encoding principle and similar to that for the one-out-of-ten system shown in Fig. 2·12, can be devised without difficulty. The use of a two-out-of-five system would have provided a degree of security against error which, while not as high as that provided by the bi-quinary system, would have been acceptable, but it would have required more complicated checking circuits, conversion to and from the decimal system would have been less trivial, and the adding circuits would have been more complicated. These disadvantages were

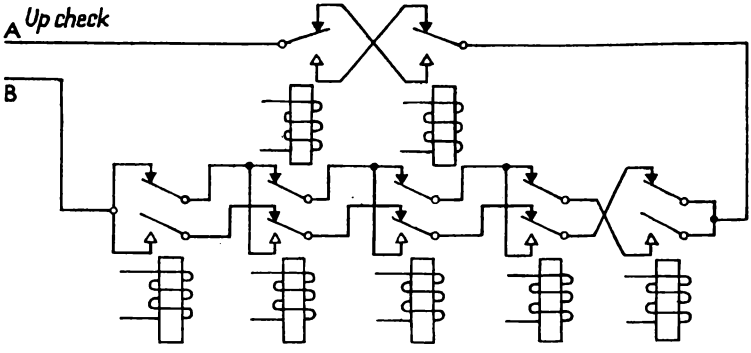


Fig. 4·1.—The 'up-check' circuit.

held to outweigh the advantage of the two-out-of-five system of requiring two fewer relays per decimal digit than the bi-quinary system.

The method used for checking can be understood by reference to Fig. 4·1, in which only those contacts on the relays which are used for checking purposes are shown. The circuit *AB* is not complete unless one relay, and only one, of each group is operated. For an error to occur in transmission or storage it is necessary that two compensating failures of an opposite character should occur; one relay which should not be operated must become operated, and one relay which should be operated must fail to be operated.

The circuit just described gives what is called an 'up-check', which indicates that a number is being stored in the register. There is another circuit (not shown in Fig. 4·1), wired through independent contacts on the relays, which is complete only when all the relays are in the unoperated condition; this is known as the 'down-check' and indicates that the register has been cleared. When a number is being transferred from one register to another, the procedure is first to clear

the second register and then to set up connexions which cause the transfer to take place. The closing of the down-check circuit indicates that the first of these stages has been completed, and is made use of to operate the relays which initiate the second stage. The completion of this stage is indicated by the closing of the up-check circuit, and this in turn is used to initiate the next step in the subsequent action of the machine. Thus, to quote Alt, 'There is nothing in the machine to synchronize the operation of its various parts; each relay operation engenders the next, just as fast as relays will click.' Thus any failure in the machine which gives rise to a meaningless setting of a register, an impossible configuration in the adder, or some similar state of affairs, causes the machine to stop, since the corresponding up-check circuit will not be completed.

It has been remarked that when an alarm occurs on a certain operation there is a good chance that if a second attempt is made the operation will be performed correctly. The Bell Model VI has been designed with a 'second trial' feature, whereby if an up-check circuit fails to be completed, then the whole operation is carried out again. Experience has shown that this feature makes long periods of uninterrupted computer operation possible.

The Bell Computer Model V

The Bell Model V installations each comprise two computers which can, if necessary, be connected together and run as one. Each computer has 15 registers, designated by the letters A to O, in its main store, and 14 other registers associated with the arithmetic and control sections of the machine and not, therefore, available to the programmer. Numbers are represented in floating decimal form, that is, $a \,.\, 10^{b}$, where a is a number with seven decimal digits and $-19 \leqslant b \leqslant 19$. To hold a number expressed in this form 62 relays are required; 49 are used to hold the seven digits of a expressed in biquinary notation, two are used to hold the sign, nine to hold the exponent, and two to hold the sign of the exponent. The arithmetic unit provides for multiplication, division, and the taking of square roots. There are about 9,000 relays in a complete installation and 50 pieces of Teletype equipment, such as tape readers, printers, and output punches.

Three kinds of input tape are used; they are known respectively as *problem tapes*, *routine tapes*, and *table tapes*. A problem may use up to five routine tapes, up to six table tapes, and one problem tape. When the machine was designed it was intended that the routine tapes should contain all the orders, that the table tapes should contain numerical information (in particular, tables of functions), and that the problem tape should contain initial conditions, coefficients, and other numerical information specific to the particular calculation

in hand. There was nothing in the design of the machine, however, to prevent the programmer putting some of his orders on the problem tape, putting numerical information on one of the routine tapes, or departing in some other way from the scheme laid down. Alt remarks that as time went on programmers found it more and more convenient to take advantage of this flexibility, with the result that the original distinction between the different types of tape disappeared. In particular, careful disposition of the information on the various tapes enables the time spent in hunting to be reduced to a minimum. This is an example of the fact that it is difficult for a machine designer to foresee exactly what arrangements will be most convenient in practice and that programmers will usually be eager to take advantage of the greatest degree of flexibility a machine provides.

An auxiliary store based on the use of punched paper tape is provided to supplement the small amount of storage available in the main store. It consists simply of a standard teletype perforator by means of which the programmer can cause information to be transferred from the main store to paper tape. This tape passes directly into a Teletype tape reader modified to enable the tape to move backwards as well as forwards. Bins for tape are fixed on both sides of the tape-reader so that a large amount of information can be stored and hunted for when required. An automatic stop is provided to prevent the tape being damaged if a number is called for before a sufficient length of tape has accumulated between the punch and the tape-reader.

Orders are punched in a simple form of three-address code in which algebraic symbols are used to indicate the operation to be performed. For example, an order calling for the number in register C to be subtracted from that in register A, and for the result to be put in register B, is punched as

$$AC - CC = B$$

followed by a symbol indicating the end of the order. This order will also cause registers A and C to be cleared when the operation is completed. If it is required to hold the number in register C (for example) instead of clearing it, the order must be punched as

$$AC - CH = B.$$

The use of letters instead of numbers to designate registers enables the same symbol to be used both for a register and its contents, and this, combined with the use of algebraic signs, makes the orders appear very simple and straightforward. If, at a certain point in the programme, it is necessary to subtract a constant from the content of register A, the numerical value of the constant can be punched directly on the tape after the minus sign, provided that certain indi-

cating symbols are also punched. Similarly, in any other order, one of the letters standing for a register can be replaced by a number. Constants in formulae can thus be handled very conveniently in this machine; in most machines it is necessary first to place a constant in a storage register before it can be brought into the calculation.

The machine is rather slow in operation. An addition (three-address) takes about 0·3 sec, a multiplication about 1 sec and a division about 5 sec. However, about 2 sec is taken to read the order for each of these operations from the programme tape, and, since the reading of one order can overlap the execution of the previous order, the speed of the machine is determined by the speed at which orders can be read. Punching a number on the output tape takes about 3 sec, but, since reading and the execution of the next order can take place while this operation is being completed, machine time is not usually wasted by reason of the slowness of the output punch. Hunting for numbers or orders on a tape is naturally time consuming. About $\frac{1}{2}$ sec is taken for each number or order passed during the hunting process, but, if the nature of the problem allows it, hunting for numbers can take place while the machine is computing. Special equipment is provided to enable values of the following functions to be obtained rapidly: $\sin x$, $\cos x$, arc $\tan x$, and (in one installation) $\log x$ and 10^x.

In addition to the arithmetic checking described above, the machine is equipped with circuits which give a degree of protection against mis-reading of the programme tape. Each type of character which forms part of an order punched on the tape is distinguished by having a different number of holes; for example, the 15 letters A to O are all represented by rows of holes in which there are three holes and three blanks. Each character in an order is tested, after being read, to verify that it belongs to the correct type according to its place in the order. If it does not, an alarm is given.

The Bell Computer Model VI

This machine was the last to be constructed by Bell Telephone Laboratories, and is installed in their own building at Murray Hill, N.J. Its main features are similar to those of Model V with a number of minor improvements, for example, the provision of 10 significant decimals in each number instead of seven, and the provision of two independently controlled storage tapes instead of one. Two major advances have also been made. One of these is the second trial feature which has already been mentioned. Earlier Bell machines would pass to a fresh section of the problem, or to a new problem already loaded on spare input channels, when a check failed, but the automatic repetition of individual operations was a new and important feature of Model VI.

The other major advance was the provision of a system whereby subroutines could be wired semi-permanently into the machine and called for by code groups punched on the order tape. A storage device developed for use in an automatic message-accounting system in a telephone exchange is used. This device depends on the fact that if a pulse of current (obtained by discharging a condenser) is passed through an insulated wire, threaded through the centre of a coil, a sufficiently high voltage can be induced in the coil to trigger a cold-cathode gas tube. The gas tube can in turn actuate a relay which locks itself in through a pair of contacts and records, for as long as is required, the information that the gas tube has been struck.

There are four vertical columns of coils; the first contains one coil for each storage register in the machine, the second contains coils corresponding to the basic operations performed by the machine—that is, add, subtract, multiply, &c.—and the third and fourth are similar to the first. If it is desired to wire in a subroutine whose first order calls for the number in A to be multiplied by that in B, and the result to be put into C, an insulated wire is taken from a socket on the left, passed through coil A in the first column, through the coil corresponding to multiplication in the second, through coil B in the third, through coil C in the fourth, and finally connected to another socket on the right. Other orders of the subroutine are similarly wired in, the wires starting from consecutive sockets on the left and terminating in the corresponding sockets on the right. There is provision for using as many as 20 wires in a subroutine when required. When the subroutine is called in, a condenser is discharged through the wire corresponding to the first order. This order then becomes recorded on the relays actuated by the gas tubes and is executed by the machine as if it had been read from the order tape. A condenser is then discharged through the wire corresponding to the second order of the subroutine, and so on. When all the orders have been executed, control is returned to the order tape if the subroutine had been called in from there. It might be, however, that the subroutine had been called in from another subroutine, in which case control would be returned to that subroutine. Provision is made for four levels of subroutines, and any subroutine can call in any other subroutine at a lower level than itself.

Up to 200 subroutines can be wired on the machine at any one time. Some subroutines, such as those for calculating elementary functions and for the multiplication of complex numbers, are likely to be left there permanently; others may be put on for specific problems and removed afterwards. Not only does the use of wired-in subroutines simplify the work of the programmer, but it also speeds up the calculation, since much reading of orders from tape—which is a slow operation—is eliminated.

THE HARVARD COMPUTER MARK II

The Harvard relay computer, which was completed in 1948, forms an instructive contrast to the Bell machines, since it could hardly be more different either in the philosophy underlying its design or in the details of its construction.

The action of the Harvard machine is synchronized with the opening and closing of pairs of contacts operated by cams which rotate at a constant speed. It is a feature of the design that the making and breaking of any circuit carrying current is performed by one of these pairs of contacts and not by contacts carried on relays. The cam-operated contacts are designed to withstand the arcing which always takes place when an inductive circuit is broken and, since these contacts are located in a group, they can readily be kept under observation for maintenance purposes. The relay contacts themselves are not subject to damage by arcing and oxidation, since they do not break current. There are in all about 800 pairs of cam-operated contacts opening and closing at different times during the operating cycle of the machine.

In addition to ordinary spring-loaded relays the machine contains 'latched' relays which have two stable positions and two operating coils. They can be moved from one position to another by passing a current for a short time through the appropriate coil. These relays are used extensively in the store, since they can hold information without the consumption of power. All relays, latched and ordinary, were specially designed for the machine. Each is fitted with six pairs of change-over contacts and takes between 6 and 10 msec to operate, although the cam-operated contacts are so timed that the minimum time in which a relay is required to operate is $16\frac{2}{3}$ msec. All relays are fitted with silver contacts. This material has the advantage that non-conducting chemical films are not easily formed on it, since both silver oxide and silver sulphide are conducting. Altogether there are about 13,000 relays in the machine.

Since the programmer is expected to provide what is needed in the way of checking, there are no checking circuits in the machine, except for a few associated with the output system.

Numbers are expressed in the machine in floating decimal form, that is $a.10^b$, where a is expressed to 10 significant figures, and $-15 \leqslant b \leqslant 15$. Each decimal digit is represented by its binary equivalent and stored on a group of four relays. Four further relays are used to store the exponent b, and two relays to store the sign of the number and the sign of the exponent respectively. There are thus 46 relays altogether in a register, and numbers are transmitted from one part of the machine to another by means of a group of 46 wires. Sixteen extra relays are associated with each register for switching purposes.

The machine is divided into two 'sides', each of which is capable of functioning as an independent machine if required to do so. Each side has a main store consisting of 48 storage registers and 12 sets of dial switches on which constants may be set. In addition, each side contains two multiplying units and an adder. Six-hole punched paper tape, of the kind used in teleprinter operation, is used both for supplying the programme to the machine and for supplying tabular data. There are two sequence control mechanisms in each half of the machine, and control may be transferred conditionally from one to another. The use of subroutines punched on endless loops of tape is therefore possible.

The machine contains no divider, but there is a unit which supplies approximate values of reciprocals. These can be refined by means of orders included in the programme, and the value of the reciprocal obtained in this way can be used to effect division. A similar system is used for the following functions: $\log_{10} x$, 10^x, $\cos x$, $\arctan x$. Special interpolator mechanisms are provided for dealing with general functions expressed in tabular form. The table, including interpolation coefficients if required, is punched on an endless paper tape which is placed on one of the interpolator mechanisms. When a value of the function is required the argument is placed in a special register and the interpolator mechanism is set in action. Four or eight consecutive numbers are then automatically read from the tape and placed in registers set aside for the purpose. These numbers can be made use of by the programme to compute an interpolated value with the aid of an appropriate interpolation formula or in any way required. There are two interpolator mechanisms on each side of the machine.

The transfer of a number from one register to another takes 33 msec, and an addition takes 200 msec. In order to achieve a high overall speed of computation it is necessary to dovetail together the various multiplications and additions that occur in the programme, so that the adder and the multipliers available on each side of the machine are all kept continuously employed. This dovetailing must be done by the programmer himself, but means are provided to prevent him from making the mistake of calling for the result of an addition or a multiplication before it is ready. One way of doing this would have been to provide mechanical interlocks. This system was not adopted, presumably because of the circuit complications it would have involved. The method now to be described was used instead.

It has been explained that the action of the machine is synchronized to a rotating shaft. One complete cycle of operation takes 1 sec and is divided into 30 steps. In each step the programmer can cause a number to be transferred from one register of the machine to another. This is done inside the machine by connecting the output of one register (the source) to a central bus, and the input of the other register

(the destination) to the same bus. The destination may either be a register in the store or a register associated with some special piece of equipment, such as an adder, a multiplier, or a printer; in the latter cases, when the transfer is complete, the appropriate action is initiated. This action then proceeds independently while the transfers called for in subsequent steps of the programme take place.

The system used in the present machine, however, differs from that used in most other machines designed on the 'source-destination' principle in that the programmer is not free to choose both the source and the destination in each step. In all cases either the source or the destination is fixed by the construction of the machine. The forms on which programmes are written give information about what is fixed and what is left to the discretion of the programmer. Such a form is reproduced in Fig. 4·2. The two columns headed 'out' and 'in' correspond to the source and destination respectively. On each line one column has printed in it the specification of a source or a destination and the other column is left blank for the programmer to fill in.

If nothing is written in the column headed 'Sign' the number is transferred from the source to the destination as it stands; by writing suitable code numbers here the programmer can cause certain simple modifications of the number, such as a change of sign, or the taking of the modulus, to be made during transfer. A code number is written in the last column when it is required that a special operation should be initiated after the transfer of the number; for example, when a number has been sent to a register associated with one of the printers, the code number written in the last column determines the style in which the number is to be printed and the layout instructions that are sent to the printer. Some operations, such as multiplication, are initiated automatically without it being necessary for the programmer to write a code number in the operations column.

Examination of the programming form reproduced in Fig. 4·2 will now show how the timing of additions and multiplications is arranged. In step 1 the programmer can send a number to the adder. In step 2 he can send a second number to the adder, which will then begin automatically to perform the addition. In step 6 the sum will be available and the programmer can transfer it to any destination he desires. Subsequent additions may be initiated after steps 7 and 8, 19 and 20, and 25 and 26. In a similar way a multiplication may be initiated by sending the factors to multiplication unit 1 in steps 3 and 4, and the product will be available in step 23. The product of factors sent to multiplication unit 2 in steps 9 and 10 will be available during step 29. It will be seen that this system of programming enables the programmer to keep the adder and the multipliers working simultaneously without any danger of an error occurring through faulty timing. On the other hand, the steps in which multiplications or additions are

initiated and those in which the results become available can be used for no other purposes, and if the programmer is not ready for those operations to be performed he must waste the corresponding steps.

PROBLEM_____ TAPE_____ SECOND_____

LINE	FORMULA	SIGN	OUT	IN	OPER.
1	x from S.R.014 to Augend R.		014	AUGEND	
2	y from S.R.035 to Addend R.		035	ADDEND	
3	a from S.R.016 to Multiplicand R.		016	MULTIPLICAND 1	
4	b from S.R.005 to Multiplier R.		005	MULTIPLIER 1	
5	a from S.R.016 to Transfer R.		016	TRANSFER	
6	$x+y$ from Sum R. to S.R.021		SUM	021	
7	$-\lvert u \rvert$ from S.R.034 to Augend R.	3	034	AUGEND	
8	T from Sw.R.107 to Addend R.		107	ADDEND	
9				MULTIPLICAND 2	
10				MULTIPLIER 2	
11	a from Transfer R. to Sign-Invert R.		TRANSFER	074	
12	$T-\lvert u \rvert$ from Sum R. to Check R.		SUM	073	
13	z from S.R.041 to Transfer R.		041	TRANSFER	
14	z from Transfer R. to S.R.022		TRANSFER	022	
15	n from S.R.001 to Transfer R.		001	TRANSFER	
16	Print arg. n and space horizontally		TRANSFER	055	02
17	Transfer r from S.R.050 to S.R.047 and		050	TRANSFER	01
18	read s from tape to S.R.050		TRANSFER	047	
19	$x+y$ from S.R.021 to Augend R.		021	AUGEND	
20	$\frac{a}{\lvert a \rvert} z$ from S.R.022 to Addend R.	4	022	ADDEND	
21	y from S.R.035 to Transfer R.		035	TRANSFER	
22	y from Transfer R. to Punch R.; punch		TRANSFER	052	01
23	ab from Product R. to S.R.023		PRODUCT 1	023	
24	$x+y+\frac{a}{\lvert a \rvert} z$ from Sum R. to S.R.036		SUM	036	
25	p from S.R.013 to Augend R.		013	AUGEND	
26	$-q$ from S.R.012 to Addend R.	1	012	ADDEND	
27				TRANSFER	
28			TRANSFER		
29			PRODUCT 2		
30	$p-q$ from Sum R. to S.R.037		SUM	037	

Fig. 4·2.—Programme form used with the Harvard Mark II computer.

One test of efficient programming with this machine is to count the number of blank lines left on the coding sheets.

Although the method described protects the programmer against errors arising as a result of faulty timing of multiplications and addi-

tions, it leaves him with the responsibility of ensuring that numbers are not sent at too short intervals to the printer or demanded at too short intervals from the tape-reader. Similarly, when he demands a sequence of four or eight consecutive numbers from an interpolator mechanism he must allow sufficient time for these values to be obtained and placed in the interpolator registers before attempting to make use of them.

From the description of the operating cycle of the machine given above, it will be appreciated that, by relay standards, Harvard Mark II is a fast machine. An ordinary telegraph tape-reader would not have been fast enough for the input of orders, and a special mechanism capable of reading 10 orders (180 holes) simultaneously three times a second was designed. After being read the orders are stored temporarily on relays until they are required.

SYNCHRONOUS AND ASYNCHRONOUS RELAY CIRCUITS

In this section an attempt will be made to give some discussion of the features and relative advantages of synchronous and asynchronous relay circuits as exemplified in the Harvard and Bell machines respectively. The examples of circuits shown in the diagrams are intended to illustrate the principles described in the text, and would not necessarily be suitable for use as they stand in a practical machine.

At first sight it would appear that use of the asynchronous system, in which the closing of each relay in the operating sequence causes the closing of the next relay, must necessarily lead to faster operation than use of the synchronous system, because no time is used up merely in providing a margin of safety. However, this conclusion would not be true, because, as will be seen from the examples given later, more relay operations are usually needed in an asynchronous device than in the corresponding synchronous device.

Fig. 4·3 shows one digital stage of a parallel accumulator for use in a synchronous binary machine. The various cam-operated contacts are denoted by W_1, W_2, W_3, W_4, and in the upper part of Fig. 4·4 the periods of the operating cycle during which they are closed are indicated by heavy black lines. During these periods the leads marked W_1, W_2, W_3, W_4 are connected to one pole of a battery (or alternative source of power) whose other pole is earthed. Addition takes place during the portion of the cycle extending from a_0 to a_5. The resident digit is stored on relay B, which is in an unoperated condition if it is storing a 0 and in an operated condition if it is storing a 1. In the latter case the relay is locked through W_4. The incoming carry digit from the less significant stage is presented to the circuit through two leads, that marked 0 being earthed if the carry digit is 0, and that marked 1 being earthed if the carry digit is 1. The carry digit generated by the circuit is similarly transmitted to the more significant

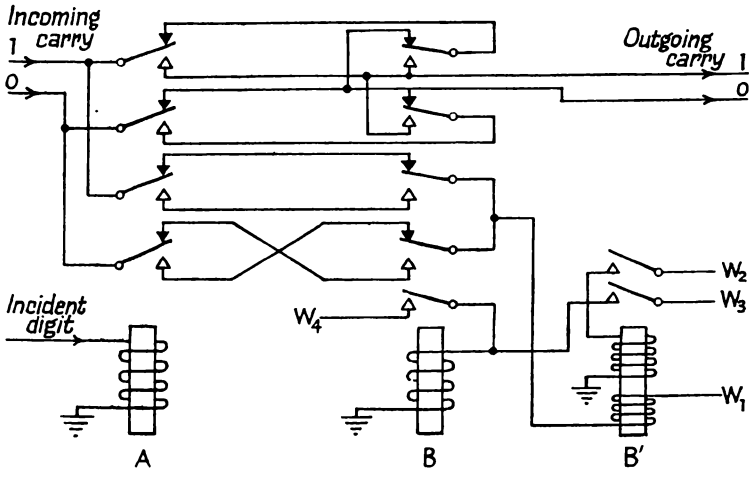


Fig. 4·3.—One stage of a parallel relay accumulator working on the synchronous system.

stage. It is assumed that at some time before a_0 the incident digit becomes set up on relay *A*. The sequence of events is then as follows:

a_0–a_1	If sum digit is 1 relay B' becomes energized through W_1 and contacts on *A* and *B*
a_1–a_2	B' (if operated) is locked through W_2
a_2–a_3	Sum digit now stored on B' is transferred to *B* by the closing of W_3 and the opening of W_4
a_3–a_4	*B* (if operated) is locked by the closing of W_4
a_4–a_5	Circuit returns to normal condition by opening of W_2 and W_3.

The working of the circuit is illustrated diagrammatically in the lower part of Fig. 4·4, which shows the time taken for the various relays to

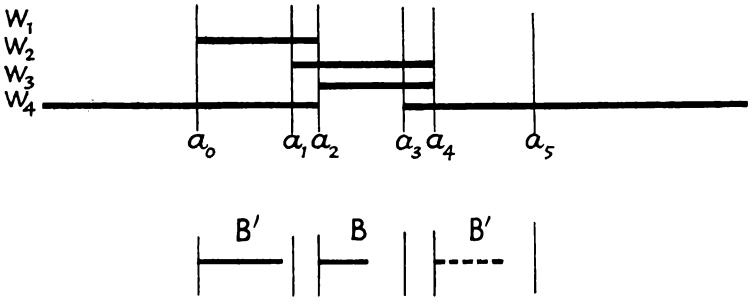


Fig. 4·4.—Timing diagram for the cam-operated contacts required to drive the accumulator shown in Fig. 4·3.

open and close when the resident digit is 0, the incident digit is 1, and the carry digit is 0. The vertical lines represent the times at which cam-operated contacts open or close, and each thick horizontal line represents the time taken for a relay to close, i.e., the time during which it is passing from the unoperated to the operated condition; a thick dotted line similarly represents the time taken for a relay to open.

The lines are of varying length, since, even if the relays have all the same operating times when they are first adjusted, variations will develop after a short period of service. It would not be excessive to allow a factor of two between the operating times of the fastest and

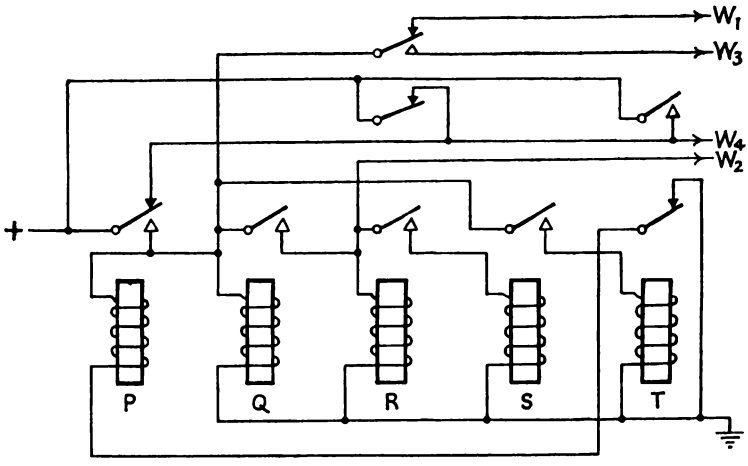


Fig. 4·5.—Chain of relays for converting the accumulator of Fig. 4·3 to work asynchronously. The action is started by energizing relay *P* momentarily through a coil additional to that shown in the diagram.

slowest relays. In the Harvard machine the time allowed for a relay to operate—this corresponds to the time between the vertical dotted lines in Fig. 4·4—is about 1½ times the time taken by the slowest relay when newly adjusted. An adequate margin of operation, even after some deterioration of the relay, is thus secured. In practice, the cam-operated contacts will not open and close exactly at the nominal times. The variation is, however, likely to be considerably smaller than the variation in the time of operation of the relays, and was neglected when Fig. 4·4 was being drawn.

The accumulator of Fig. 4·3 can be converted to asynchronous operation by providing a chain of relays to take the place of the cam-operated contacts. Such a chain, in which each relay controls the operation of the succeeding relay, is shown in Fig. 4·5, and its operation

is illustrated in Fig. 4·6. This latter figure is to be compared to Fig. 4·4. In Fig. 4·6 the distances between the vertical lines are determined by the times of operation of the relays of the control chain, *P*, *Q*, *R*, *S*, *T*, and are subject to variation according to the state of adjustment of the relays; in Fig. 4·4 the corresponding distances are determined by the cams on the continuously rotating cam shaft and are not subject to anything like the same degree of variation. It will also be noticed that the condition that the relay *B′* should not break current has been relaxed in the asynchronous case. There would be no point

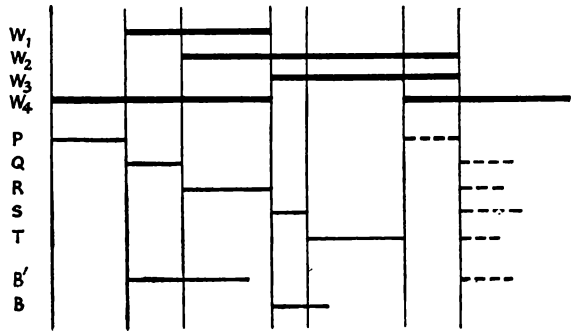


Fig. 4·6.—Timing diagram for the accumulator of Fig. 4·3 when operated asynchronously in conjunction with the relay chain shown in Fig. 4·5.

in retaining this condition as far as the relay *B′* is concerned, since the relays of the control chain, *P*, *Q*, *R*, *S*, *T*, must necessarily break current.

It will be seen that each step in the addition process involves the closing of two relays of the control chain; this is in order that the time of operation of the relays *A*, *B*, and *B′* shall be adequately covered. Even so, unless the ratio of operating times of the slowest and fastest relays can be kept to a figure appreciably better than 2 : 1, the margin of safety will be small.

AUTOMATIC ERROR DETECTION

The failure of a pair of contacts directly associated with the operation of the successive relays in the chain will cause the machine to stop. Failure of other contacts can, however, cause errors which are likely to go undetected. If extra contacts were fitted to the relays it would be possible to provide additional checking circuits. For example, a circuit could be provided which would not be complete unless relays *B* and *B′* were both operated or both unoperated. Failure of the circuit to be complete at the correct time would lead to

a stoppage. However, anything more than partial and patchy protection against errors could hardly be obtained in this way. The real difficulty in the design of circuits which are fully self-checking is that although the closing of a pair of contacts on a relay indicates that that relay has been energized, it does not follow that all other contacts on the relay have been satisfactorily closed, since some faults, such as the intrusion of a particle of dust, may affect one pair of contacts alone.

The design of asynchronous relay circuits can be carried out on a much more satisfactory basis when redundant representation is used, as in the Bell machines. In a binary machine (unless some form of weighted count is used) this implies the use of two relays per binary digit. One circuit is wired on each pair of relays so as to be continuous when both are in the released position and no digit is therefore being stored; this is the 'down-check'. There is also an 'up-check' circuit wired so as to be complete when one of the relays, and only one, is operated. With this system it is possible by suitable use of the 'up-check' and 'down-check' circuits to arrange that relays used for handling information also form part of the control chain. This is not possible where a single relay is used to store a binary digit, as can be seen by considering the transfer of a digit stored on one relay to a second relay which initially stores a 0, i.e., is in an unoperated condition. If both relays initially hold 0, the second relay does not move when the transfer operation takes place; this relay cannot, therefore, form a link in the control chain.

The use of redundant representation in a binary machine in the straightforward manner just described would involve the use of twice the number of relays that would otherwise be required. This degree of redundancy is probably higher than is necessary, and it could be reduced by arranging the binary digits in groups and associating an extra relay, or a number of relays, with each group in order to provide the required redundancy. In the case of a decimal machine working with a binary code, the grouping is provided naturally.

Redundant representation can also be used in synchronous machines, and 'up-check' and 'down-check' circuits can be fitted. It can be arranged that the machine will stop if these circuits are not continuous at appropriately chosen moments during the operating cycle. It should be noted that the 'up-check' and 'down-check' circuits are here used simply for checking purposes and that their opening and closing do not form essential steps in the chain of operation of the machine. If trouble develops in a relay associated with one of the checking circuits it is possible to disconnect that circuit. The machine is then serviceable, although of course a measure of security against error has been lost. This point is of some importance, since an objection frequently urged against the provision of checking circuits is that failures

occurring in them tend to reduce the overall reliability of the machine. The argument just given, however, applies to persistent faults only and not to transient faults.

Whenever a pair of contacts of a telephone relay close a certain amount of bouncing always takes place. Fig. 4·7 is a sketch based on an oscillogram taken to illustrate this bouncing. It shows the potential difference across a resistance connected to a battery through a pair of relay contacts which are closed and then opened. The waveform of the voltage applied to the relay coil is also shown. The relay used to obtain this oscillogram was not in particularly good adjustment, and it will be seen that contact is made and broken again a number of times before the relay finally settles down. Even with a

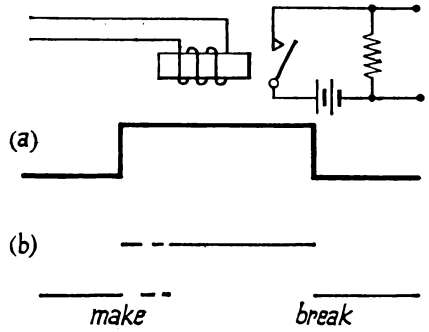


Fig. 4·7.—Sketch of an oscillogram showing relay bounce. The waveform of the voltage applied to the relay coil is shown at (*a*) and the voltage across the resistor is shown at (*b*).

relay in good adjustment one or two bounces must always be expected. When specifying the time of operation of a relay an overall figure covering all bounces should be given.

In a synchronous machine relay bounce is of no importance whatever, since the bouncing will die down in the margin of time allowed before the next step in the operating cycle is reached. Again, no difficulty is likely to be experienced on account of contact bounce in asynchronous machines in which relays with similar operating speeds are used throughout, since the period during which a relay contact bounces is always short compared with the total time the relay takes to operate. Trouble is, however, quite likely to occur if high-speed relays with an operating time of 1 or 2 msec are controlled by contacts of an ordinary telephone relay which takes, say, 15 to 20 msec to operate. The high-speed relay may open and close several times before the slow-speed relay controlling it has settled down. It is useful to bear in mind, when designing circuits in which relays of different

operating speeds are used, that bounce occurs only during make and that the break is always clean.

A special word of warning may be issued in connexion with the use of mixed circuits containing both relays and flip-flops. Here the disparity of speed is very great indeed, and special measures may have to be taken to eliminate trouble due to bounce. This disadvantage, together with the fact that the electronic circuits are likely to be disturbed by interference due to arcing at relay contacts—in practice this can never be entirely suppressed—unfortunately spoils a number of otherwise very attractive schemes.

It is sometimes necessary to design relay circuits for controlling solenoid-operated mechanical equipment, such as tape readers, punches, and printers. Such circuits are required not only in the input and output systems of relay computers, but also in auxiliary tape preparation and editing equipment used in conjunction with automatic computers generally. The operation of a solenoid usually takes much longer than the operation of a relay, and the time is inclined to be variable, not only on account of variations in spring tension, but also on account of variations in the mechanical load, for example, variations in tape friction in the case of a tape reader. It is, therefore, very desirable that methods of interlocking based on the use of 'up-check' and 'down-check' circuits should be used. For this purpose, each solenoid armature must be fitted with contacts which close when it is in its unoperated position and with other contacts, preferably entirely distinct, which close when it is in its operated position.

The use of slugged relays which open or close slowly is not usually to be recommended for computing circuits, since the operating time of these relays is highly variable. The same applies to the use of relays whose action is retarded by connecting large condensers across them. Again, the use of relays in which certain contacts open or close before other contacts is not generally to be recommended on account of the difficulty of keeping such relays in good adjustment.

Motor-operated tape readers, punches, and printers are generally to be preferred to the solenoid-operated variety, since their time of operation and the acceleration of their moving parts are fixed within close limits. Motor-operated devices are, moreover, usually very simple to connect to other apparatus; all that is required to set them in motion is the energizing of a clutch magnet, while the end of the operation may be indicated by the opening or closing of a pair of cam-operated contacts.

CHAPTER 5

STORAGE

A DISCUSSION of the design of the various components which make up an automatic calculating machine can well begin with the store, since the design of other parts of the machine is conditioned very much by the type of store it is decided to use.

There are a variety of physical principles, or properties of materials, which may be exploited in the construction of a large-capacity store. It is not necessary that the storage provided should be of an intrinsically permanent nature, although if it is not, some arrangement has to be made whereby the information is regenerated, that is, periodically rewritten. For a particular physical principle to lend itself to the design of a practical high-speed store, however, not only must it provide, at a reasonable cost, storage for a large number of binary elements, but it must also be possible readily to obtain access to the various elements at will. Thus the design of a large-capacity store presents a switching problem as well as a storage problem.

ULTRASONIC STORAGE

The first practical large-capacity store to be developed was the one using ultrasonic delay tanks, filled with mercury, which has already been described briefly in Chapter 2. Here access to the required digits is obtained by opening the output gate at the correct time.

The velocity of sound in mercury is about 1·5 mm/μsec. The longest tanks which can conveniently be handled are about 1½ metres long and give a delay of about 1 msec. The amount of information which can be stored in such a tank depends on the interval between the successive clock pulses (the pulse interval). For example, the pilot model of the ACE uses a pulse interval of 1 μsec and stores 1,024 digits in a tank 1½ metres along. Other machines have been built with somewhat shorter tanks; these give a correspondingly shorter average access time. The SEAC, for example, uses tanks about 50 cm long which, with a pulse interval of 1 μsec, give a capacity of 360 binary digits and an average access time of 168 μsec. On the other hand, if a shorter pulse interval is used, a correspondingly larger storage capacity is obtained with a given length of tank; difficulties associated with the design of the electronic circuits associated with the tank, however, set a lower limit to the pulse interval which may be used if an adequate operating margin is to be obtained. An example of a machine which uses a very short pulse interval is the UNIVAC, which operates with a pulse interval of 0·5 μsec. It should be noted that the

temperature coefficient of velocity of sound in mercury at room temperature is 0·04 per cent per degree Centigrade, and that temperature control to within a fraction of a degree is therefore necessary. The accuracy required depends on the product of the length of the tank and the clock pulse repetition frequency, that is, on the maximum number of pulses which can circulate in a tank. The problem of temperature control might, therefore, become acute if an attempt were made to use both very short pulses and a very long tank.

Mercury Tanks

The tanks may be constructed of stainless steel as in the UNIVAC, of mild steel as in the pilot model of the ACE and in the EDSAC, or of glass as in the EDVAC and SEAC. The acoustic impedances of quartz and mercury are approximately equal (their ratio is about 3 : 4), and this is the principal reason why mercury has been generally used as the acoustic medium. Mercury has the further advantage of being a conductor of electricity, so that there is no difficulty in applying the pulses to the crystal. Further, its temperature coefficient is fairly small and does not vary greatly over the range of temperatures which are likely to be used. It is necessary to secure good acoustic contact between the mercury and the quartz crystals. This can be done by using clean mercury and clean polished quartz crystals, when it will be found that the mercury will wet the surfaces of the crystals. An alternative method, which does not necessitate the use of polished crystals, is to fill the tank first with alcohol. When the mercury is poured in the alcohol is displaced, leaving a thin film on the surface of each crystal. Since alcohol wets both crystal and mercury this film acts as a bridge for the passage of acoustic energy. It is too thin for the rather severe mismatch between the acoustic impedances of quartz and alcohol (about 15 : 1) to be of importance. If alcohol is not used a film of air is formed between the mercury and the unpolished crystals and, in this case, the mismatch is so severe that the film, in spite of its thinness, has the effect of reducing transmission virtually to zero.

Commercial alcohol contains water, which tends to cause rusting if mild steel is used for the tanks. In addition, alcohol has a deleterious effect on rubber washers. For these reasons a mixture of glycerine and water (in the proportions 2 to 5), to which a small quantity of sodium benzoate was added to prevent rusting, was used instead of alcohol when filling the ultrasonic tanks used in LEO. These tanks are similar in design to those used in the EDSAC.

Some of the acoustic energy which travels the length of the column of mercury and impinges on the receiving crystal is reflected. This energy travels back to the transmitting crystal, where some of it is again reflected and finally arrives at the receiving crystal after having

made three passages of the column. In the case of a tank 1½ metres long this does not cause any trouble, since the attenuation in the mercury is sufficiently great (in the case of the EDSAC about 14 db for each passage) for the amplitude of the pulse to be reduced to a negligible level by the three passages of the column. In short tanks, however, where the column of mercury is only a few centimetres long, many successive reflections can take place before the amplitude of the pulse becomes negligible, and the difficulty is a serious one. It may be overcome in either of two ways. One is to design the crystal mounting in such a way that the amplitude of the reflected pulse has a small value. In the EDVAC this is done by supporting the crystal at its periphery and arranging for it to be in contact with the mercury on two sides. Energy falling on the crystal is then transmitted through the crystal into the mercury behind; very little reflection takes place because the crystal is half a wavelength thick and is, in any case, well matched acoustically to the mercury. After travelling a few centimetres in the mercury behind the crystal the pulse reaches an irregular plug at the end of the tube and its energy is scattered. The alternative method used in the EDSAC is to place an obstruction in the mercury column between the crystals. This has the effect of scattering part of the energy on each passage so that the indirect pulses are attenuated in relation to the direct pulses. If this method is used the crystals can be placed in dry contact with the rear electrode. The small air gap that then exists, except at a few points of contact, prevents energy from passing into the electrode, with a corresponding gain in transmission of 3 db for each crystal.

The ultrasonic tanks themselves form one part only of the store. There must also be electronic equipment for maintaining circulation of pulses and for permitting the transfer of information to and from the store. Successive degeneration of the pulses at each passage of the tank is prevented by replacing each pulse, after it has emerged from the tank and been amplified, by a fresh pulse from a clock pulse generator, in the manner which has already been described in Chapter 1 (p. 32). A diagram, more detailed than that shown in Fig. 1·19, of the circulating system used in the EDSAC together with the input, output, and clear gates is shown in Fig. 5·1. Before being applied to the transmitting crystal the pulses are modulated on to a carrier of frequency 13·5 Mc/s. Stores have been built in which modulation on to a carrier was dispensed with and the pulses were applied direct to the transmitting crystal. Since, however, the ultrasonic tank will not pass signals down to zero frequency, the pulses emerging from the receiving crystal are then badly distorted, having a negative as well as a positive swing. On the other hand, if a carrier is used together with an amplifier of sufficient bandwidth, the rectified pulses can be made very similar to the original pulses entering the modulator. This

greatly facilitates the design of a reliable clock gating circuit with wide tolerance in respect of variations in the circuit parameters and in the temperature of the tanks.

There is some latitude in the choice of the carrier frequency. It is easier to obtain the required bandwidth in the amplifier if a fairly high frequency is chosen; on the other hand, the attenuation of ultrasonic waves in mercury increases with the square of the frequency, and this is a reason for keeping the frequency low in the case of very long tanks. However, in shorter tanks the use of a high frequency helps to keep the amplitude of multiply reflected waves to a low value.

In the EDSAC the ultrasonic tanks are located in a thermostatically controlled oven and connected to the electronic circuits by means of co-axial cables matched at both ends. Some of these cables are as

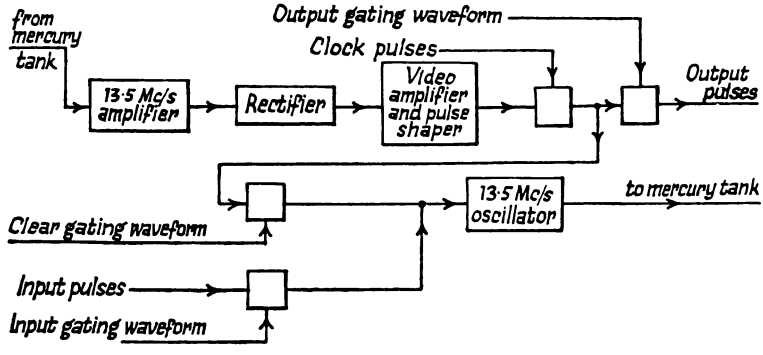


Fig. 5·1.—Block diagram of the circulating system used in the ultrasonic store of the EDSAC.

much as 10 ft long. In certain other machines the amplifiers and other components associated with the recirculation are placed close to the tanks so that the connexions to the crystals can be kept short. A disadvantage of this arrangement is that the power dissipated by the electronic circuits is given off as heat in close proximity to the tanks, so that the design of the temperature control system requires more care than if the tanks are located well away from the circuits. On the other hand, the power losses associated with matching are avoided and the system is more easily screened to prevent the intrusion of stray pulses from outside. Such interference is most likely to enter at the input to the amplifier, where the level is lowest, but, fortunately, even here the level is not low enough to make the provision of adequate screening at all difficult. In the EDSAC, for example, the signal level at the input to the amplifier is about 10 mV at 70 ohms.

The pulses emerging from the amplifier must have a greater amplitude than and be wider than the clock pulses they are used to gate. It is possible to use clock pulses of the same width as the pulses applied to the modulator and to rely on the lengthening of the pulses which is

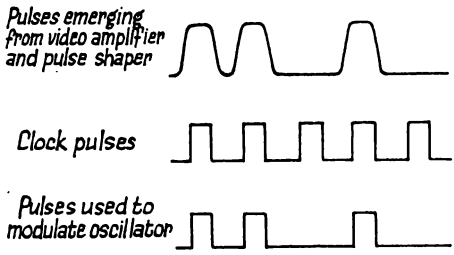


Fig. 5·2.—Waveforms for the system shown in Fig. 5·1.

bound to take place during transmission through the tank and amplifier, especially if the overall bandwidth is somewhat restricted. This, however, is likely to produce a system in which the margin of operation is poor and dependent on the tuning and gain of the amplifier. A better method is to use the pulses emerging from the amplifier to gate very short clock pulses. Those pulses which pass through the

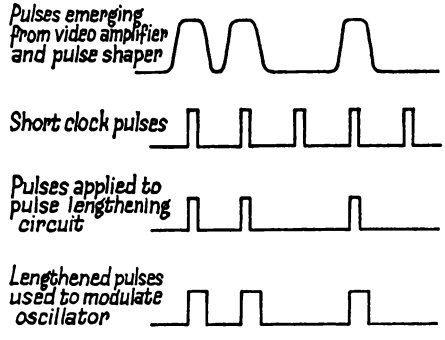

Pulses emerging from video amplifier and pulse shaper
Short clock pulses
Pulses applied to pulse lengthening circuit
Lengthened pulses used to modulate oscillator

Fig. 5·3.—The use of short clock pulses in the circulation system of an ultrasonic store.

gate are lengthened in a separate circuit to make them suitable for applying to the modulator. Figs. 5·2 and 5·3 illustrate the two systems. It will be clear that the second system enables a considerably improved tolerance in respect of temperature variation of the mercury in the tank (which causes the pulses emerging from the amplifier to be shifted in phase in relation to the clock pulses) to be achieved. The EDSAC originally used the first system, but was modified in 1951

to use the second system. A system equivalent to the second is used in the SEAC.

Although the use of reasonable care is necessary in filling mercury tanks, attainment of the highest degree of chemical cleanliness does not seem to be necessary. The design of the tanks should be such that no substance which reacts appreciably with mercury is present. In the EDSAC rubber washers were used and, while no trouble was experienced for two, or perhaps three, years, after that time it was necessary to empty the tanks, clean them, and refill them.

Modulation of the pulses on to the carrier should be 100 per cent, that is, the carrier should be completely suppressed during the interval between the pulses. One method of achieving this is to use an oscillator which is normally so heavily damped as to be quiescent. When a pulse is applied the damping is removed and the circuit bursts into oscillation. This system is used in the EDSAC. Alternatively, use can be made of a continuously running oscillator, the output of which is gated with the applied pulses. There can either be a separate oscillator on the electronic chassis associated with each tank or a central oscillator to serve all chassis. The former system is perhaps to be preferred, since each chassis can then be adjusted and tested as a self-contained unit.

Solid Ultrasonic Delay Media

Soon after it had been shown that a practical store could be constructed with mercury tanks, attempts were made to use solid substances, such as quartz, for the ultrasonic medium. These attempts met with indifferent success. The difficulty is that in elastic solids two or more types of wave, with different polarizations, can be propagated. If these waves travel with different velocities, it is necessary to ensure that one only is excited by the transmitting element and that no excitation of any other wave takes place during the course of transmission, either as a consequence of reflection from a boundary, or as a result of inhomogeneity in the medium. The technical difficulties have now been overcome in the case of fused quartz, largely as a result of the development of more satisfactory methods of attaching the crystals used for transmission and reception, and it is possible to construct very satisfactory delay units using this material.

In 1951 it was found that very good results could be obtained by using nickel wire as the ultrasonic medium and making use of its magnetostrictive properties to excite and detect the waves. All that is necessary in order to excite a wave is to apply a current pulse to a small coil wound round the wire. This form of electro-acoustic transducer has the advantage of exciting a wave of very nearly pure longitudinal form. It can be used with quite thin wires (0·005 in.) or with thin-walled (0·002–0·003 in.) tubes in which transverse waves are

heavily damped. A similar coil connected to an amplifier is used at the receiving end, a small permanent magnet being placed near by in order to provide 'magnetic bias'. The system is shown diagrammatically in Fig. 5·4.

The wire can be bent, provided that the radius of curvature is large compared with the diameter of the wire; this is a matter of some importance, since the velocity of propagation of sound in solids is much higher than in liquids, and about 5 metres of nickel wire are required to give 1 msec delay time, compared with $1\frac{1}{2}$ metres of mercury. By coiling the wire into a flat spiral of, say, 30 cm radius, each turn being supported lightly in three or four places only, a compact delay unit can be constructed. Unwanted reflections from the ends can be quite easily prevented by clamping the wire between pads of rubber, or some other absorbing material, or by embedding it in a tube of grease. The requirements for temperature control are less exacting in the case of nickel than mercury, since the coefficient of variation of

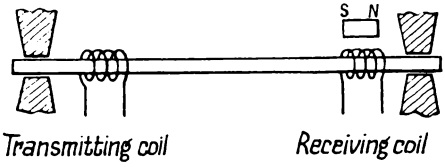


Fig. 5·4.—A magnetostrictive delay unit.

velocity with temperature for pure nickel is only 0·014 per cent per degree Centigrade, and alloys with a negligible temperature coefficient can be made.

The pulses are applied directly to the transmitting coil without being modulated on to a radio-frequency carrier. Storage systems working with a pulse interval of 3 μsec have been constructed. It is found that for a given delay the overall loss is somewhat less than in a mercury tank in which the pulses are modulated on to a carrier. A smaller proportion of the loss is contributed by attenuation of the ultrasonic waves during transmission as distinct from losses in the transducers, and much longer delays would, therefore, be possible with nickel wire or tube than with the conventional system using mercury.

The passage of a current through the transmitting coil creates (virtually instantaneously if the wire is thin) a stress in that part of the wire which is in the field of the coil. Let the time taken for a compressional disturbance to travel a distance in the wire equal to the length of the coil be t_0. Consider the effect of passing through the coil a pulse of current whose duration is long compared with t_0, but whose times of rise and fall are short compared with t_0 (see Fig. 5·5 (*a*)). A

pair of similar compressional disturbances travel along the wire in opposite directions. One is damped when it reaches the end of the wire, while the other travels the length of the wire and enters the receiving coil. The form of this latter disturbance is shown in Fig. 5·5 (*b*), in which the displacement of a particle of the wire from its initial position is plotted (for a fixed time) against distance along the wire. The compression in the wire at any point is obtained by differentiating the curve with respect to distance, the result being as shown

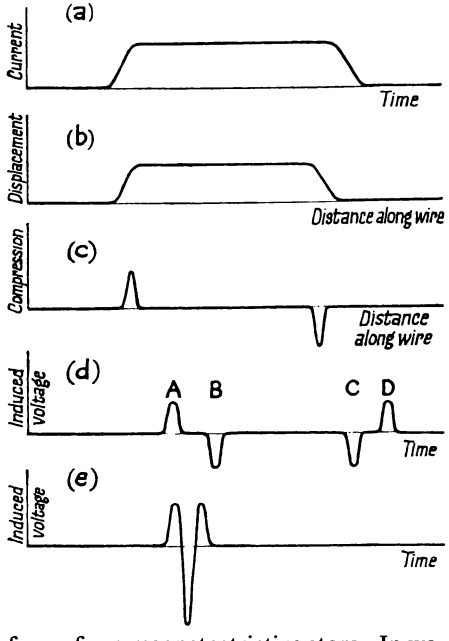


Fig. 5·5.—Waveforms for a magnetostrictive store. In waveform (*d*), peak A is formed when leading edge of compressional pulse enters coil; peak B is formed when leading edge of compressional pulse emerges from coil; peak C is formed when trailing edge of compressional pulse enters coil; peak D is formed when trailing edge of compressional pulse emerges from coil.

in Fig. 5·5 (*c*). The disturbance thus consists of two compressional pulses, one positive and one negative, separated by a distance determined by the duration of the original current pulse and the velocity of propagation.

When the first compressional pulse enters the receiving coil, the permeability of the wire inside the coil is changed as a result of the compression, and, since a polarizing field is present, this results in a change of flux. An e.m.f.—positive, say—whose magnitude depends

on the rate of change of mean permeability of the wire within the coil, is therefore induced during the time that the first compressional pulse is entering the receiving coil and a negative e.m.f. during the time that it is leaving; similarly, a negative e.m.f. is induced when the second compressional wave is entering the coil and a positive e.m.f. when it is leaving. There are thus four peaks in the induced voltage waveform as shown in Fig. 5·5 (*d*). If the size of the coil is suitably chosen in relation to the length of the pulse, the peaks marked B and C can be made to coalesce. If, preserving this relationship, the pulse length is then made short, the waveform of Fig. 5·5 (*e*) is obtained. This is the waveform obtained under operating conditions. The central negative-going portion is selected by means of a discriminating circuit and used to control the passage of a clock pulse in the manner described earlier in connexion with the use of mercury tanks.

ELECTROSTATIC STORAGE

It is convenient to group together under this heading a number of storage devices in which numbers are represented by distributions of electric charge on a surface inside a vacuum tube. Early in the development of digital computers, it was realized that it should be possible to store information in the form of a charge distribution on the screen of an ordinary cathode-ray tube by writing on it with a suitably modulated beam of electrons, and references to the possibility of constructing a system working on these lines were made at the Moore School of Electrical Engineering in 1946. The distribution of charge on the surface of the screen would not, of course, be permanent, and some method of regeneration would be necessary; in 1946 it seemed that it would be necessary to use two tubes and to shuttle the information continually from one to the other.

Williams and Kilburn's Store

The first practical system of electrostatic storage using an ordinary cathode-ray tube was described by Williams and Kilburn in 1948, and this system has since been widely used. In it a binary number is represented by a sequence of dots and dashes—representing 0s and 1s respectively—written on the screen by applying a timebase to one pair of deflecting plates and a suitable switching waveform to the modulating electrode of the electron gun. In addition to producing transient flashes of light the electron beam sets up a distribution of electrons on the surface of the fluorescent screen, the distribution having one form where a dot has been written and another form where a dash has been written. If, after an interval of time sufficiently short for no appreciable leakage of charge to have taken place, the line is scanned again, this time with the beam turned on continuously, the current which at any instant flows into the screen will depend on whether the beam is

passing over a place where a dot was written on the previous sweep or over a place where a dash was written. The flow of current may be detected by observing the induced current which flows to or from a piece of metal gauze placed in front of the screen, in contact with the outside surface of the tube.

Williams and Kilburn showed that it is not necessary to keep the reading beam on continuously, since it is possible to determine from the nature of the response whether a given symbol is a dot or a dash when the first part only of that symbol has been scanned. Sufficiently early information is, in fact, obtained to make it possible for the symbol to be rewritten as a dot or a dash in the same sweep. Numbers

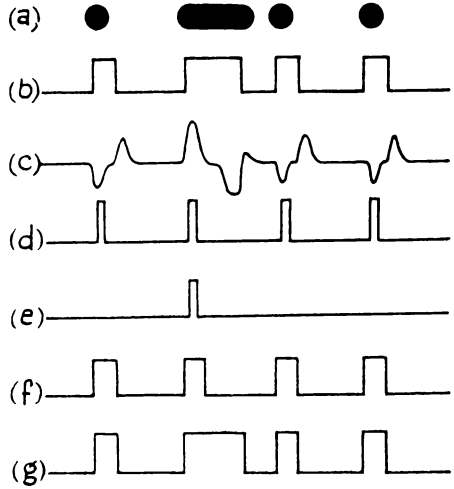


Fig. 5·6.—Waveforms for Williams electrostatic storage system.

stored on a tube can therefore be regenerated without it being necessary to transfer them to another tube in the process. An explanation, in general terms, of how it comes about that such early information can be obtained is as follows. During writing, the influence of the beam extends over a greater area than that on which electrons are actually impinging, since secondary electrons are given off by the screen, and some of these find their way on to adjacent areas. In particular, this influence extends in a direction opposite to that in which the beam sweeps, with the result that, during the subsequent reading scan, an indication is obtained as to whether the beam was switched on or off at a particular point before that point is actually reached.

The actions of reading and regeneration may be understood in greater detail with the aid of Fig. 5·6. (*a*) shows the pattern of dots

and dashes which it is supposed are being written on the screen. (*b*) shows the waveform applied for this purpose to the modulating electrode of the electron gun. (*c*) shows the waveform picked up from the gauze attached to the front of the tube when the pattern written on the screen is subsequently scanned with the beam continuously on. It will be seen that, for each character (dot or dash), the early part of this waveform is negative-going for a dot and positive-going for a dash. It is only this part of the waveform which is made use of, and in order to isolate it the signal is gated with the waveform shown at (*d*), the result being shown at (*e*). It makes no difference to the waveform obtained after gating if, instead of being scanned with the beam continuously on, the pattern is scanned with the beam

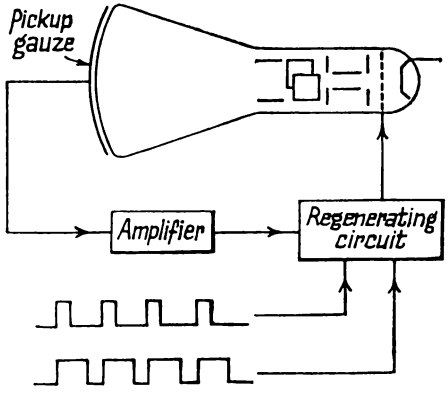


Fig. 5·7.—Block diagram of Williams electrostatic storage system.

switched on and off by the waveform shown at (*f*). Such a beam writes a series of dots. Regeneration can now be achieved by lengthening the dot in waveform (*f*) every time a positive pulse is observed in waveform (*e*). There is plenty of time to do this, since the pulses in waveform (*e*) occur during the early parts of the dots in waveform (*f*). The resulting waveform applied to the modulating electrode is shown at (*g*) and, since this is a replica of (*b*), the original pattern is rewritten and true regeneration is obtained. A block schematic diagram of the regenerating system is shown in Fig. 5·7. Naturally, in a practical store, means must be provided for interrupting the regeneration and writing a new pattern when it is required to change the information stored. For a full explanation of the physical processes which determine the nature of the waveform picked up by the external electrode mounted on the face of the tube the reader is referred to papers by Williams and Kilburn.

The signal level at the pick-up gauze attached to the front of the cathode-ray tube is extremely low, and a gain of some 60 db is necessary in the amplifier. The system is, therefore, sensitive to interference at this point, and if satisfactory operation is to be obtained careful attention must be paid to screening. An interfering pulse from outside, with a duration of only a few microseconds, can easily cause a permanent change in the information recorded in the store.

Although it is quite possible to make use of ordinary cathode-ray tubes in a storage system, it is usually found that in any given batch of tubes there are a number with small defects which make them unsuitable for storage purposes, and selection is necessary. It is better to use tubes which have been specially developed for the purpose. Perhaps the most serious defect to which ordinary tubes are subject is the occurrence of flaws on the surface of the fluorescent screen. These can take the form of minute pinholes in the fluorescent coating which leave the bare glass exposed, or of minute portions of some extraneous substance embedded in the coating. A common source of the latter is the aquadag which is often sprayed on the inside of the glass to give (after heat treatment) a conducting carbon deposit surrounding the screen. In addition, other irregularities on the screen, of a less obvious kind, can give rise to flaws. In tubes specially intended for storage the number and size of the flaws can be reduced by taking precautions to ensure cleanliness. Strangely enough it has not so far proved possible to find any substance which is an appreciable improvement on the ordinary phosphor used for coating the screen of cathode-ray tubes.

It would appear that, whatever precautions are taken, there are bound to be some irregularities in the screen which can be detected by observing the current flowing to the pick-up gauze when the beam is swept across the screen. It is desirable to choose conditions of working in which the effects of these irregularities are as small as possible and then, for a given system, to establish limits which tubes must pass before they are acceptable.

The effect of irregularities on the screen is minimized if the tube is run with a comparatively low accelerating voltage. This is because the difference between the secondary emission ratios of the impurities and of the phosphor—impurities tend to have a lower secondary emission ratio—are less pronounced at the lower voltages.

To obtain good focus at a comparatively low accelerating voltage a specially designed gun is needed. Since cheapness and ease of production in quantity are not prime requirements in a tube intended for storage purposes, a more elaborate electrode structure and closer assembly tolerances than those found in ordinary cathode-ray tubes are possible. The optical response of a phosphor falls off very rapidly as the intensity of bombardment is reduced and, in an ordinary

cathode-ray tube, the focusing of the electron beam often appears very much sharper than it actually is. As a result it sometimes happens that tubes which give a good spot on the fluorescent screen are quite useless for storage purposes on account of an undue number of electrons reaching the fringe areas in the neighbourhood of the focus. This difficulty does not occur in specially designed tubes fitted with high-grade electron guns.

The residual irregularities on the screen of a good tube are quite small in scale, and their effects can be minimized by using a system in which a greater area of the screen is used to record each binary digit than in the one so far described. Such a system is the 'defocus–focus' system. Here, when a 0 is to be recorded the beam is switched on in a defocused condition and, after a short interval, switched off again; when a 1 is to be recorded the beam is again switched on for a short interval in a defocused condition, but is then focused, so as to cover a smaller area of the screen, and held for an instant in this condition before being switched off. In this way it is possible to make use of a larger proportion of the total available area of the screen than when the dot–dash system is used, and the effect of small scale irregularities in the screen is reduced accordingly.

The Williams type electrostatic store can be used either in the serial mode, in which it was originally demonstrated, or in the parallel mode. In both systems the dots and dashes (or the defocused and focused spots) which represent binary digits are arranged in a series of parallel rows on the screen. In the serial system the digits recorded along each horizontal line represent the successive digits of a single binary number, or perhaps of two binary numbers; when reading or writing, the beam is first switched to the correct line (or to the beginning of the correct section of the line) and then made to scan the line so that the digits of the numbers are read or written serially. Interleaved with such 'action' scans there are 'regenerative' scans which serve the purpose of maintaining the information recorded in the store. Normally there is one regenerative scan to each action scan. The action scans take place along lines selected according to the requirements of the programme, whereas the regenerative scans are arranged to cover systematically all the lines on the screen in the course of each regenerative cycle.

In the machines built by Ferranti Ltd and based on the design of Williams and Kilburn, each cathode-ray tube in the high-speed store contains 32 lines, each line holding two numbers of 20 binary digits. Regeneration of all the lines takes place approximately once every 30 msec on the average. Since regeneration is so frequent no very high degree of stability is called for in the timebase circuits and a slow drift of the raster over the surface of the screen does not matter. This facilitates, in particular, the design of the digital-to-analogue con-

verter which accepts, in binary form, the number of the line to be scanned and gives out a voltage proportional to the magnitude of that number for application to the vertical deflecting plates of the cathode-ray tube.

In a fully parallel store it is necessary to provide as many tubes as there are binary digits in the numbers to be stored; one digit of each number is held in each tube. For reading and writing the electron beam must be directed to the correct spot on the tube instead of to the beginning of a particular line as in the serial system. Two digital-to-analogue converters are thus necessary in this system, one for each direction of motion of the spot. Regeneration takes place systematically, spot by spot, at least one spot being regenerated between each reading and writing operation.

Although there are no fundamental differences between the serial and parallel modes of operation as far as the recording and reading processes are concerned, more is asked of a parallel store in regard to the holding of information in the intervals between regeneration. Slight damage is always done to the information stored in a particular spot when a reading or writing operation is performed on an adjacent spot. If a given spot is subjected to this treatment a large number of times before the information stored in it is regenerated, that information may be destroyed. Now, in a parallel machine with 1,024 spots on the tubes, it might conceivably happen that information was read from spots adjacent to a given spot as many as 1,024 times before the given spot was regenerated, and situations in which this occurs more than 100 times are not uncommon in practice. On the other hand, in a serial machine with 64 lines on a tube, the maximum number of times a line adjacent to a given line can be read or written before the given line is regenerated is only 64, and, moreover, because of the way in which information is distributed on the various tubes composing the store, the danger of a critical situation occurring in a programme is remote.

In order to provide a measure of the goodness or badness, in relation to this particular defect of a parallel store, it is convenient to define a quantity known as the *read-around* ratio. This is defined as the average number of times information can be read out of a spot without disturbing the information stored in a neighbouring spot. The read-around ratio is found to vary from point to point on the screen of each tube, being greatly affected by flaws or slight irregularities in the surface. A read-around ratio equal to the theoretical maximum which can be attained in a programme is naturally to be desired, but a store in which the read-around ratio is everywhere greater than 50 is found to be workable in practice. If the read-around ratio falls below 10 it is difficult to use the machine at all for serious calculations. Since the places where poor read-around ratios occur are

determined by the positions of flaws and irregularities on the various tube screens, if a machine is not very good from the point of view of read-around ratio, small movements of the rasters due to the spot positioning circuits can profoundly affect its serviceability.

Other Electrostatic Storage Systems

A system of electrostatic storage was developed in the digital computer laboratory at M.I.T. As in the Williams system binary digits were stored by means of charges located in a rectangular array on a screen placed in a vacuum, and access to any particular digit was obtained by deflecting a cathode-ray beam so as to strike the corresponding element of the screen. A special tube, which was not a modification of an ordinary cathode-ray tube, was developed for the purpose. The system was used for several years in Whirlwind I, and gave satisfactory performance having regard to its experimental nature. However, it has since been replaced by the matrix storage system described later in this chapter, and is not likely to find further application. Nevertheless, the operation of the tube depends on an interesting physical principle and is worth a short description.

The M.I.T. storage tube used a system of continuous regeneration depending on the fact that an insulated conductor whose surface has

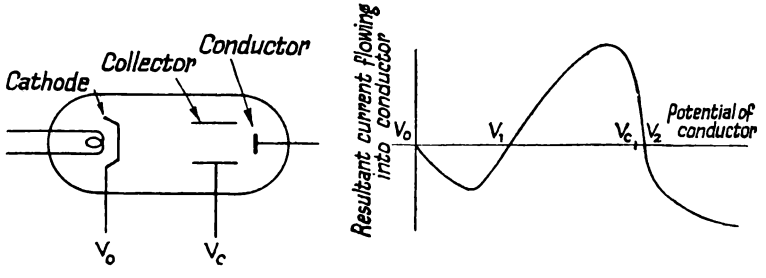


Fig. 5·8.—Current flowing into a conductor placed in a beam of electrons in a vacuum.

secondary emitting properties will, when placed in a vacuum and exposed to a beam of electrons, take up one or other of two possible equilibrium potentials. That this is so may be seen by considering Fig. 5·8, the right-hand part of which shows the resultant current flowing into the conductor (that is, the difference between the number of secondary electrons leaving per second and the number of primary electrons arriving) plotted against the potential of the conductor. The secondary electrons eventually find their way to a collector which is at a potential V_c. V_c is greater than V_0, the potential of the cathode. Let us suppose first that the conductor is at some potential lower than

V_0. No electrons from the cathode will be able to fall on the conductor, and its potential will gradually rise on account of leakage of electrons through the glass of the envelope to the collector. As soon as the potential of the conductor reaches V_0, electrons will fall on it and prevent its potential from rising any higher. The conductor will thus have reached an equilibrium potential which will be slightly less than V_0 on account of the finite velocity with which electrons are emitted by the cathode. If the conductor has its potential forcibly and steadily increased above V_0 it attracts more and more electrons. At the same time secondary electrons begin to be emitted. At first the ratio of the number of secondary electrons emitted to the number of primary electrons striking the conductor is small, but as the velocity of impact of the primary electrons increases, so does this ratio, until a point is reached at which there are as many secondary electrons emitted as there are primary electrons incident. This is shown in Fig. 5·8, where the curve of current flowing to the conductor crosses the axis at a potential V_1. If the potential of the conductor is further increased the ratio of secondary to primary electrons continues to increase. The resultant current to the conductor thus becomes positive and increases with the potential of the conductor. When the potential of the conductor reaches the potential of the collector, however, the resultant current drops sharply, since the secondary electrons, which up to this point have been finding their way to the collector, now begin to fall back on to the conductor. The curve in Fig. 5·8, therefore, crosses the axis again at V_2 and finally becomes parallel to the axis at a level determined by the total emission from the cathode.

It is easily seen that V_2 is a voltage of stable equilibrium for an insulated conductor, whereas V_1 is a voltage of unstable equilibrium. There are consequently two and only two stable potentials which a conductor, placed in an electron beam in a vacuum, may take up, one being approximately the potential of the cathode and the other approximately the potential of the electrode which collects any secondary electrons emitted by the conductor. The two stable potentials may be used in a variety of ways to effect the storage of a binary digit.

The M.I.T. storage tube was fitted with two electron guns. One was of the type used in an ordinary cathode-ray tube and focused a beam of electrons on to the storage surface. This gun was provided with deflector plates in the usual way and was used, in conjunction with a signal plate electrostatically coupled to the storage surface, to effect reading and writing. Except during reading and writing the beam from the gun was suppressed. The other gun was of a very simple structure and produced a continuous diffuse beam of electrons which covered the whole storage surface. This beam had the effect,

as just explained, of maintaining the elements of the storage surface at one or other of two distinct potentials which were taken to correspond to 0 and 1 respectively. During the writing operation the action of this beam was overridden by that of the more powerful beam from the first gun.

Another electrostatic storage device using the same regenerative principle was the Selectron developed by J. Rajchman in the Research Laboratories of the Radio Corporation of America. In this tube the storage elements were small brass eyelets supported in a mica plate. Access to the elements was obtained not by deflecting a beam but by allowing electrons to pass through apertures placed in front of the various elements. By the time development of the tube was completed, however, other storage systems had come into use, and Selectrons did not therefore find the wide application which might otherwise have been expected for them. A detailed description of their construction will not, therefore, be given here.

Picture Storage Tubes

Various tubes have been constructed for the purpose of storing television images or the displays used in radar and similar systems. So far, however, none of these tubes have proved suitable for use as storage devices in a digital computer. This is perhaps not surprising in view of the different requirements which the two types of tube must be designed to meet. A picture-storage tube must provide storage for a very large quantity of information—several hundred times as much as the digital storage tubes just described—but some degree of interaction between adjacent spots can be tolerated. Further, the access requirements are quite different. For these reasons it is doubtful whether a tube suitable for both requirements will ever be produced.

MAGNETIC STORAGE

This section will deal with the important group of storage devices which depend on magnetic remanence in ferro-magnetic materials. In the first type to be discussed the magnetic material moves relatively to a coil of wire used for writing and reading. An example is the auxiliary store of the EDVAC, which makes use of magnetic wire and was referred to briefly in Chapter 1. Other systems of the same type make use of paper, or, more commonly, cellulose acetate, tape which has been sprayed with magnetic material. Another storage device, working on the same principle, is the magnetic drum; this is a continuously rotating drum made of non-magnetic material with a thin layer of magnetic material deposited on its cylindrical surface.

The principle by which digital information is recorded on and read from a moving magnetic medium is the same as that used in magnetic

sound recording. There are, however, a number of practical differences arising from the fact that in a digital storage system two levels of recording only are required instead of a linear response over a range of amplitudes.

In the following explanation it will be assumed for simplicity that information is being recorded on a thin wire made of some magnetizable material with good remanence—steel piano wire for instance—although what is said will be equally applicable to a thin coating of magnetic material carried on a tape or a drum. Fig. 5·9 (*a*) shows the wire passing in close proximity to a recording head, which consists of a coil of wire wound on a laminated soft-iron core with a small gap. To begin with it will be supposed that the wire is in a demagnetized condition and is at rest. A pulse of current having the waveform shown at (*b*) is now passed through the coil. A magnetic field is built up

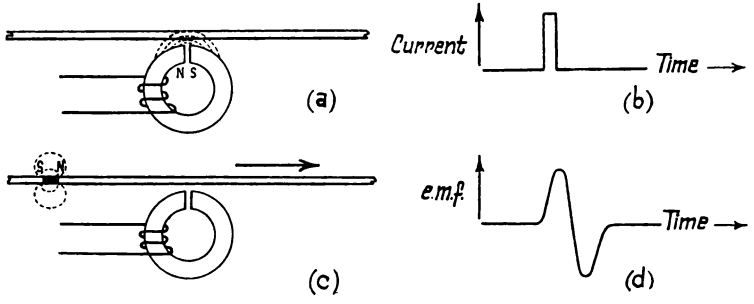


Fig. 5·9.—Recording on magnetic wire.

across the gap, and a leakage field, with lines of force similar to those sketched in the diagram, spreads through the wire and causes it to become magnetized. If the current passed through the coil is of sufficient intensity, the leakage field saturates the short section of the wire immediately opposite the gap and, when the field dies down at the end of the pulse, this section retains some of its magnetization. The wire further from the gap, however, returns very nearly to its original demagnetized condition. The result is that a short magnetic dipole is left impressed on the wire. Some self-demagnetization of this dipole takes place, especially if the diameter of the wire is comparable with the length of the dipole.

Normally, during recording, the wire is in motion. If the duration of the pulse and the time constant of the head are both sufficiently short for there to be no appreciable movement of the wire during the time the magnetic field is changing, this motion can be neglected. If, however, the wire moves an appreciable distance after the field has built up and before it falls to zero, the recorded dipole will be drawn

out and its length may be many times that of the gap in the recording head.

The reading process has now to be considered. Fig. 5·9 (*c*) shows the dipole approaching the reading head, which is similar in construction to the recording head. The lines of force carried with the dipole enter the core and give rise to an e.m.f. in the coil. At any instant this e.m.f. is proportional to the rate of change of flux and builds up to a maximum in one direction as the dipole approaches the head, falls to zero, and builds up to a maximum in the other direction, finally falling again to zero as the dipole recedes from the head. The waveform of the induced e.m.f., which is shown in Fig. 5·9 (*d*), is thus balanced about zero, and its amplitude is proportional to the speed of the wire during reading.

The waveforms obtained from the reading head as a result of the passage of dipoles which were originally recorded on wire by pulses

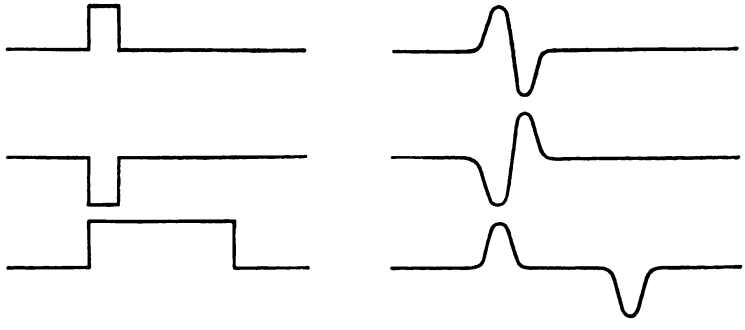


Fig. 5·10.—Signals obtained from a reading head for different forms of recording current waveform. The recording current waveforms are shown on the left and the signals obtained on the right.

of opposite polarity are compared in Fig. 5·10, which also shows the waveform read back from a dipole long compared with the gap in the reading head. These diagrams will make it clear that the waveform obtained from the reading head is in no sense a replica of, or a near approximation to, the waveform applied to the recording head. Instead, it has the general form of a differentiated version of that waveform, although this is not a true description, since the time scale depends on the dimensions of the gap in the recording head and the speed of the wire during reading, as well as on the duration of the pulse used for recording. Also, if the wire has been properly saturated during recording, the amplitude of the output signal depends on the speed of the wire and on the properties of the reading head rather than on the amplitude of the recording pulse.

Representation of Digits

Methods of recording digits on a moving magnetic medium may be divided into two principal classes: (i) those which require the recording (and in due course reading back) of synchronizing pulses on a separate track from that on which information is recorded, and (ii) those in which the recording of synchronizing pulses is unnecessary. Methods of the first class may be used with magnetic drums and magnetic tape, but cannot be used with magnetic wire, since this can accommodate one track only.

The simplest method of recording (of the first class) is to record on the tape or other magnetic medium a dipole for a 1 and nothing at all for a 0. The current in the writing head is then normally zero, and swings positive when a 1 is being recorded. The waveform of the writing current is shown in Fig. 5·11, together with the waveform of

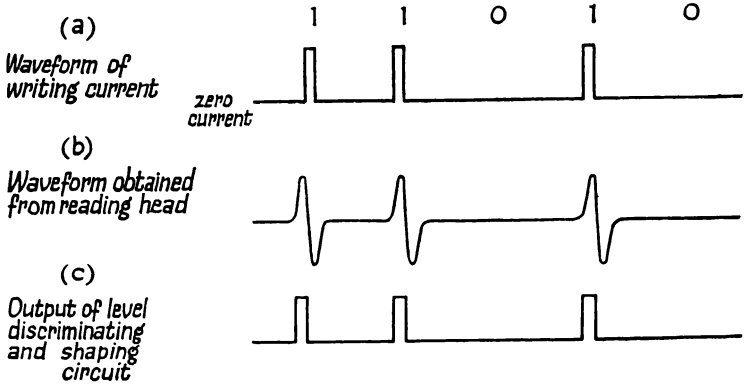


Fig. 5·11.—The dipole method of recording.

the signal obtained from the reading head. If this signal is passed through a level discriminating and shaping circuit, a waveform resembling that of the original writing current can be obtained. This is also shown in the diagram. This system of recording has the disadvantage that the medium must first be demagnetized before new information can be recorded on it. Demagnetization requires the use of an alternating field, which must be of high frequency if demagnetization of rapidly moving magnetic material is required.

An improvement on the system just described is obtained by using the writing waveform shown in Fig. 5·12. Sufficient current normally passes through the writing head to saturate the magnetic material in the negative direction. When a 1 is being written the current is reversed and made to swing sufficiently positive to saturate the tape in

the other direction. 1s are thus recorded as positive dipoles embedded in tape which is otherwise uniformly magnetized in the negative direction. The waveform of the signal obtained from the reading head is the same as before (see Fig. 5·11), except that the amplitude

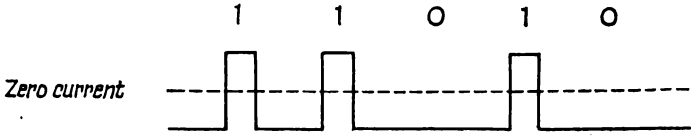


Fig. 5·12.—Alternative waveform of writing current for the dipole method of recording.

is doubled. No demagnetization is necessary, new information being written directly over information no longer required.

In both the methods just described a 1 is indicated by the presence of a pulse in the waveform obtained from the reading head and a 0 by the absence of a pulse. In order to make it possible for this waveform

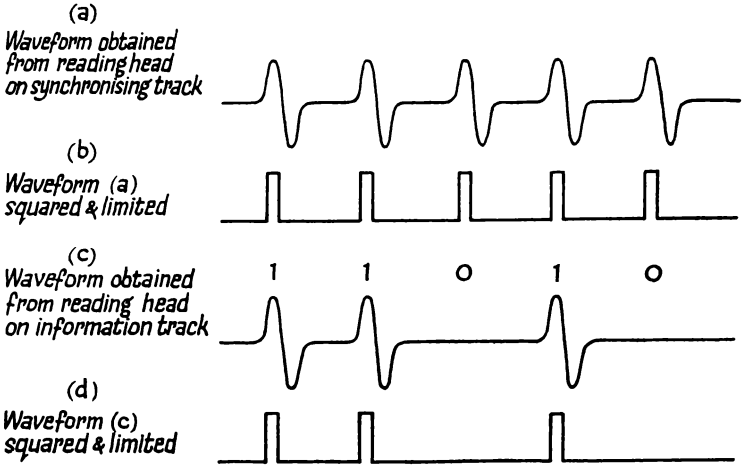


Fig. 5·13.—The dipole method of recording with a synchronizing track.

to be interpreted without ambiguity some reference signal must be provided for timing purposes. The usual method is to record, on a separate track on the drum or tape, a sequence of equally spaced synchronizing pulses (see Fig. 5·13). These are read back, amplified, and shaped by means of circuits similar to those used for the information pulses. The resulting waveform consists of a set of equally spaced pulses. These can be used to interpret the output from the informa-

tion channel; a pulse on both channels indicates a 1, and a pulse on the synchronizing channel alone indicates a 0.

The method just described will be referred to as the *dipole method*; an alternative is one sometimes known as the *non-return-to-zero method.* The waveforms for this method are shown in Fig. 5·14. As before, the writing current swings from a negative to a positive value, but, when a sequence of 1s is being recorded, it remains constant at

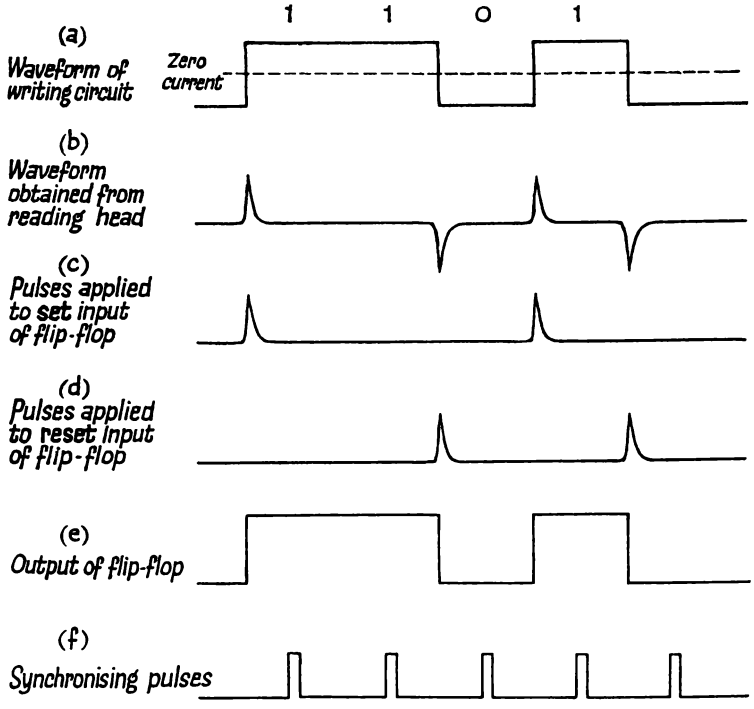


Fig. 5·14.—The non-return-to-zero method of recording.

its positive value until the end of the sequence is reached, when it returns to its negative value. The waveform obtained from the reading head consists of a series of short pulses; a positive-going pulse indicates the beginning of a sequence of 1s, and a negative-going pulse indicates the end of such a sequence. The two sets of pulses are separated, and the positive-going pulses are applied to one input of a flip-flop while the negative-going pulses are applied, after reversal, to the other input. The waveform of the positive-going output from the flip-flop is then of the same form as the signal originally applied to the recording head. With this method also a synchronizing signal

recorded on a separate track is necessary. One method which has been used successfully is to record a square wave of half the digit period. The signal obtained on reading back can be used to produce pulses occurring once in each digit period, as shown at (*f*) in Fig. 5·14. The flip-flop output can then be interpreted in relation to these pulses; when a synchronizing pulse occurs a 1 is indicated if the flip-flop output is high and a 0 is indicated if the flip-flop output is low. The non-return-to-zero method allows a greater recording density than the dipole method, since there is never more than one transition from a positive to a negative value of current, or vice versa, between the recording of successive digits.

A phase-modulation system has been described by Williams, Kilburn, and Thomas. In this system when a 1 is being recorded the

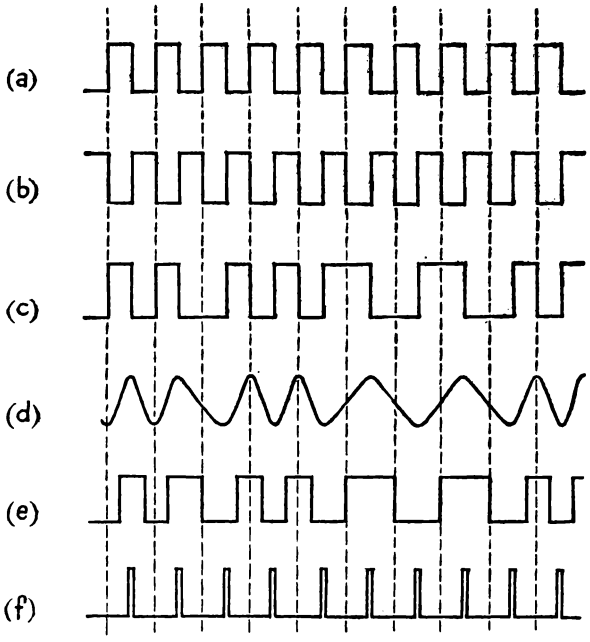


Fig. 5·15.—The phase-modulation method of recording.

writing current is positive during the first half of the digit interval and negative during the second half; for a 0 it is negative during the first half and positive during the second half. When a 0 follows a 1 or vice versa the current does not change at the moment of transition. Examples of various recording waveforms are shown in Fig. 5·15. A sequence of 1s is represented by a square wave of digit frequency

(shown at (*a*)), and a series of 0s by a similar square wave displaced 180° in phase (shown at (*b*)). A waveform which represents a number containing both 1s and 0s is shown at (*c*).

Reading circuits very similar to those used for the non-return-to-zero method could be used for reading back information recorded by the phase-modulation method. The above authors, however, use a simpler method which takes advantage of the fact that a relatively restricted bandwidth, which does not extend to zero frequency, may be used to transmit phase-modulated signals. That this is so may be seen by noting that the highest and lowest fundamental frequencies which can occur in a repetitive waveform are f and $\frac{1}{2}f$ when f is the digit frequency; the former corresponds to a sequence of 1s or a sequence of 0s, and the latter to a sequence of alternate 1s and 0s. Williams and Thomas use a circuit in which the reading head, or rather its associated transformer, is broadly tuned to the frequency f, in their case 100 kc/s. They then find that a sequence of 1s (or a sequence of 0s) gives rise, on reading, to an approximately sinusoidal waveform of frequency f, while a sequence of alternate 1s and 0s gives a waveform, also approximately sinusoidal, but of frequency $\frac{1}{2}f$. If the tuning and damping of the head circuit are suitably adjusted, the amplitude of these two sine waves can be made approximately equal. Fig. 5·15 (*d*) shows (somewhat idealized) the waveform obtained when the number consisting of mixed 1s and 0s is read back. If this is amplified and the amplitude limited in both directions, the waveform shown at (*e*) is obtained. This is very similar to that of the original writing current, but the transitions are delayed. If this waveform is gated with the synchronizing pulses shown at (*f*), it is possible to obtain a waveform in which a 1 is represented by the presence of a pulse and a 0 by the absence of a pulse. In order to eliminate the delay, Williams, Kilburn, and Thomas use a reading head slightly in advance of the head used for writing.

Although the successful operation of the phase-modulation system calls for careful adjustment of the tuning and damping of the circuit containing the reading head, it is found that once made these adjustments are stable, and that the system is very satisfactory in operation. Oscillograms of waveforms obtained in practice are reproduced in reference 104.

A method referred to as the *polarized-dipole method* may be used when no synchronizing channel is available. Here the material is initially in a demagnetized condition and is magnetized positively for a 1 and negatively for a 0. On reading back a 1 gives a signal which swings first positive and then negative and a 0 gives a signal which swings first negative and then positive. Since both 1s and 0s are represented by definite signals, instead of by the presence and absence of signals respectively, no synchronizing channel is necessary with

this method. Various circuits may be used to distinguish 1s from 0s; an example, together with explanatory waveforms, is given in Fig. 5·16. When this method of recording is used the magnetic material must be demagnetized before new information can be recorded.

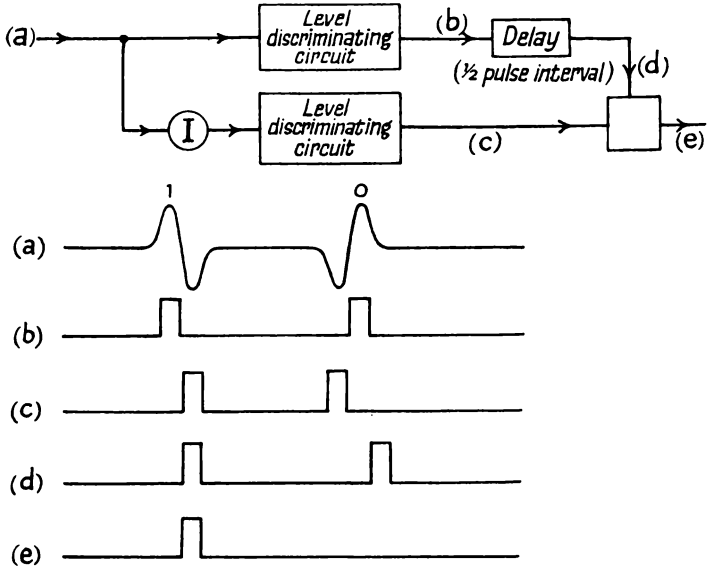


Fig. 5·16.—Separation of 1s from 0s in polarized dipole method of recording when no synchronizing track is used. Waveforms at (*a*), (*b*), etc., are given in the lower part of the diagram.

The method of recording just described may also be used in conjunction with a synchronizing channel, although, when such a channel is available, one of the methods described earlier would appear to be preferable. If the magnetic material is not properly demagnetized before a dipole is written, the waveform obtained from the reading head is likely to be unsymmetrical and even to exhibit three or four peaks instead of two. For this reason any attempt to avoid the preliminary demagnetization by arranging for new information to be superposed exactly on old information (as is theoretically possible if a synchronizing track is available) is not to be recommended.

Magnetic Drums

A magnetic drum can be used conveniently either as the main store or as the auxiliary store of an automatic computing machine. If it is used as the main store the machine will be comparatively slow unless use is made of methods of minimum-access coding.

Drums are usually made of brass or bronze. They may be sprayed with a preparation of red or black oxide of iron, or, alternatively, they may be nickel-plated. Both methods may be used with success, although in each case there is a certain amount of technical 'know-how' to be acquired before consistently good results can be obtained. In particular, it is important to note that commercial nickel-plating baths usually contain impurities which render them unsuitable for plating magnetic drums.

Drums are normally run with the heads mounted as close to the surface as is possible without touching. Generally speaking, the heads used must have a wider bandwidth than the heads used in sound-recording on account of the very short pulses which must be

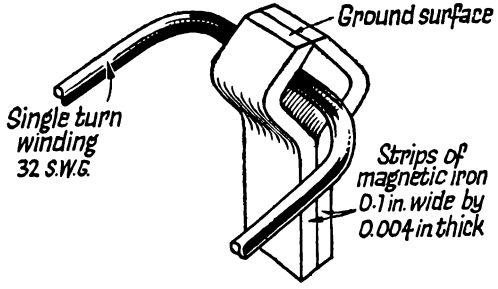


Fig. 5·17.—Sketch of a miniature head suitable for use with a magnetic drum.

handled. At the present time the design of heads is largely an empirical subject. Miniature heads suitable for use with a drum may be made by bending strips of high-permeability magnetic material about 0·004 in. thick. An example is illustrated in Fig. 5·17. The winding consists of a single turn, and the action depends on the residual air gap found along the line of contact of the two strips. Heads of this general type were first designed by Dr A. D. Booth. The head shown is designed for use with a transformer, but high-impedance miniature heads may also be constructed. These are to be preferred, if the non-return-to-zero method of recording is used, on account of the difficulty of designing a transformer with an adequate low-frequency response characteristic.

A small gap is essential if a high density of recorded information is to be obtained, but the nature of the field at some distance from the gap is also important, since it determines the effect, if any, which the recording of a digit has on digits which have previously been recorded. The stray field from the coil itself can sometimes cause trouble in this respect, particularly if the recording current is high, and this may put an upper limit on the intensity of the recording current which can be

used. For this reason, the coil should, if possible, be wound round a part of the head remote from the gap.

The width of the head determines the width of the track used, and will vary from design to design. Reducing the width of the head enables more tracks to be accommodated on a drum of given width, but, on the other hand, reduces the signal–noise ratio. Heads as narrow as $\frac{1}{32}$ in. have been proposed, but this is probably rather small; on the other hand, $\frac{1}{10}$ in. is quite sufficient.

The density with which digits can be packed on a magnetic drum depends on a number of factors. As already pointed out, the use of either the non-return-to-zero method or the phase-modulation method of recording enables a higher packing density to be obtained, for the same degree of reliability, than the dipole method. Increasing the diameter of the drum increases the length of the track, but the number of digits which can be stored does not increase in proportion, since the mechanical tolerances on the drum have to be relaxed and the spacing of the heads from the drum increased. Similarly, making the drum smaller reduces the amount of information which can be stored on a single track, but again not proportionally, since finer tolerances and smaller heads can, up to a point, be used. A recording density of 100 digits per inch (40 per centimetre), or even slightly more, may be used with complete success on a drum about 10 in. in diameter. If this density is much exceeded, however, the danger of intermittent errors occurring increases rapidly. If a short access time is required, the diameter of the drum should be kept as small as possible, since the maximum speed at which a drum can be rotated without risk of its breaking up is proportional to the peripheral speed. Most of the early successful magnetic drums ran at speeds of not more than 3,600 r.p.m., but high-speed drums running at 12,000 r.p.m. or more are now being introduced.

The same closeness of packing cannot be expected when access is required to individual digits on each track as when digits are always recorded or read in sequences containing a fixed number of digits. This is because in the former case a certain amount of space must be left between each digit and its neighbours in order to allow a degree of latitude in the timing of the writing waveform. Where access is required to single digits a system of recording in which the digits are spaced from one another (such as that shown in Fig. 5·11) is therefore to be preferred to systems such as those shown in Figs. 5·14 and 5·15, in which adjacent digits are joined together. This may be made clear by reference to Fig. 5·18, which shows the effect of altering two digits (chosen at random) recorded on a track. The upper part of the diagram shows, in idealized form, the magnetization of the material when the digits 110110 are recorded by the dipole method in which 1s are recorded as positive dipoles embedded in material that is otherwise

uniformly magnetized in a negative direction. Underneath is shown the waveform of the recording current necessary to change the third digit into a 1 and the fifth digit into a 0. Note that the current pulse used for recording a zero is longer than that used for recording a 1. The waveform of the recording current is shown as being slightly early compared with the previously recorded digits. It will be seen that the only effect which this has on the resultant state of magnetization of

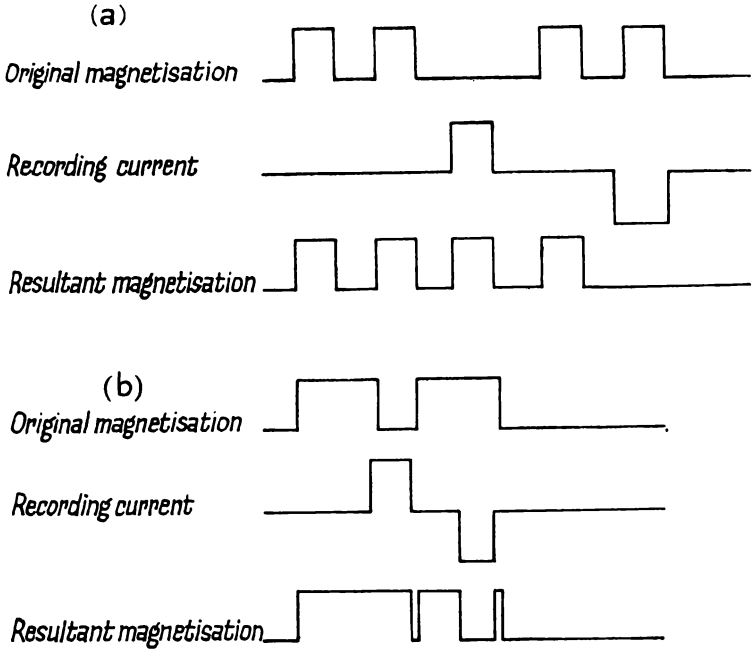


Fig. 5·18.—The effect of a small error of timing when altering individual digits in: (*a*) the dipole method, and (*b*) the non-return-to-zero method of recording. In each case the number 110110 is originally recorded and the third and fifth digits are altered.

the material is that the third digit is recorded slightly early. This need cause no difficulty, if the reading circuits are suitably designed, on account of the clear space which exists between digits in this system. In the lower part of the diagram similar information is given for the non-return-to-zero method. Here the effect of the waveform of the recording current being early is to produce 'spikes' on the curve showing the final state of magnetization of the material. These spikes would be fatal to the correct operation of the system. Formation of similar spikes was avoided in the polarized-dipole method of record-

ing by making negative-going recording pulses longer than positive-going pulses; this cannot be done in the case of the non-return-to-zero method on account of the symmetry which exists between the recording of 1s and 0s. Although given here in relation to drums, considerations of somewhat similar type apply in the case of other storage systems in which digits are recorded linearly on a surface; an example is the Williams cathode-ray tube store described earlier.

It has been pointed out that for the same standard of operating margin a greater storage capacity can be provided on a magnetic drum if digits are handled in blocks. It follows that, if the obtaining of maximum storage capacity is a prime object, a drum intended to be used as the main store of a machine should be designed to work in the serial mode, numbers being converted to the parallel mode, if this is required by the design of the arithmetic unit, by means of a shift register. When the drum is used as an auxiliary store it can be operated equally well in either mode, but if it is used in the parallel mode, transfers of numbers should take place in blocks of fixed size, a space being left on the drum between neighbouring blocks.

If the recording density is somewhere between 60 and 120 digits per inch the total number of digits which can be recorded on a single track of a drum 10 in. in diameter is between 1,800 and 3,600. One may thus think in round figures of a single track as having capacity for between 50 and 100 numbers, each with the equivalent of about 10 decimals. Many tracks are, therefore, necessary in order to obtain a large storage capacity.

When the number of tracks runs into hundreds, the problem of how to switch to the selected track is a serious one. The simplest solution is to use relays, and this method has the advantage that one reading amplifier and one writing circuit only are required in a serial machine, or one for each digit in a parallel machine. The use of high-speed relays enables the switching to be performed in a few milliseconds, a time which is usually short enough, compared with the rotation period of the drum, to be acceptable. Sooner or later, however, relays inevitably give trouble, and the trouble is often of an intermittent nature. An electronic switching system is, therefore, to be preferred, but, since several valves are required for each track, the total amount of electronic equipment required in the case of a drum with a large number of tracks is considerable.

It is usually easier to provide electronic switching for reading than for writing, especially if low-impedance heads are used. A compromise which has been adopted in some machines is, therefore, to use relays for switching the writing current and electronic switches for selecting the reading head. The switching problem is eased if separate heads are used for reading and writing. Other considerations enter here, however, such as ease of mechanical adjustment, and

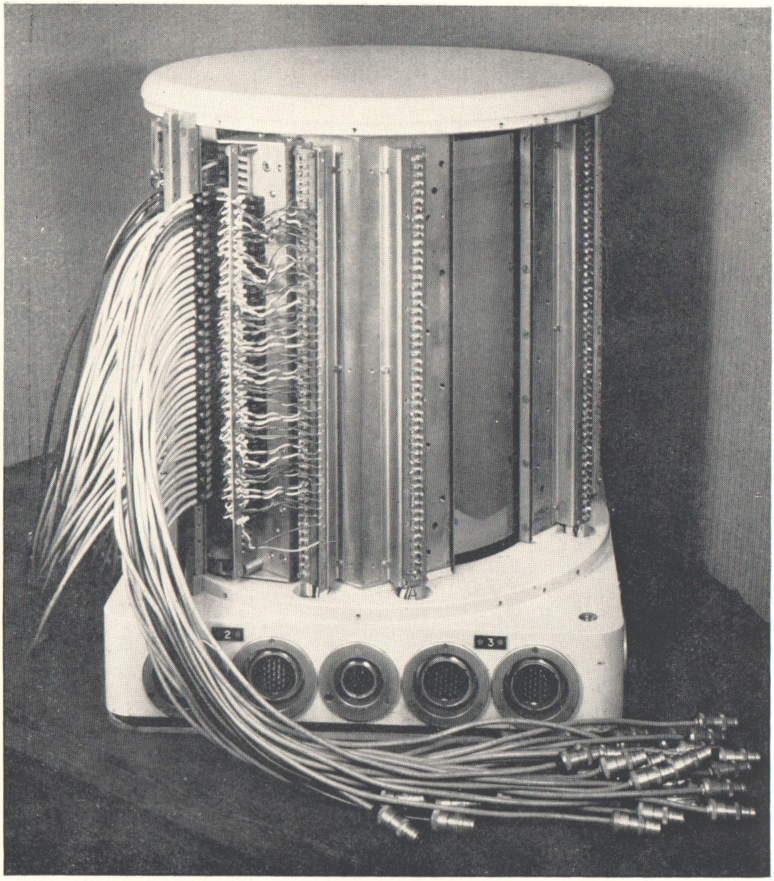


FIG. 5·19.—PHOTOGRAPH OF A MAGNETIC DRUM MADE BY FERRANTI LTD.

the phase relationship which it is desired to have between the recorded and read-back signals.

In the drum associated with the ACE (pilot model), instead of one head permanently mounted on each track there are a small number of heads mounted on a movable carriage which can be traversed in a direction parallel to the drum. The procedure for selecting a track is to switch to the appropriate head and also to traverse the carriage as a whole until the head in question reaches the required track.

In most installations the speed of the drum is not accurately controlled, and the rest of the machine is synchronized to it (in so far as this is necessary) with the aid of pulses derived from a synchronizing track on the drum. Williams and Kilburn, however, synchronized their drum to pulses derived from a crystal-controlled oscillator, which also supplied clock pulses to the rest of the machine. They were able to obtain the necessary very high degree of synchronization by means which were both simple and elegant. The system has two advantages. In the first place it is very simple to associate as many drums as are required with a single machine and no time is lost when switching over from one to another. In the second place, the clock pulses for the machine are derived from a crystal oscillator and are not as subject to jitter as they would be if they were derived from a synchronizing track on the drum; this makes it easier not only to obtain satisfactory operating margins throughout the machine, but also to achieve the maximum possible packing density of digits on the drum.

An example of a precision-built drum, small in physical size but large in storage capacity, is that made by Ferranti and shown in Fig. 5·19. This drum is nickel-plated and runs at 1,955 r.p.m. The packing density is 98 digits per inch, and this has been found by experience to be the highest density of packing which can be used if a high degree of reliability is to be obtained. Allowing for four digits between words, the drum accommodates 128 words of 20 binary digits on each track, of which there are 256. The pulse interval is 10 μsec.

Logical Designs for Drum Stores

Figs. 5·20 and 5·21 show typical logical designs for stores based on the use of magnetic drums. The store shown in Fig. 5·20 works in the parallel mode and has capacity for 1,024 numbers. Separate heads are used for reading and writing, although, in the figure, the two heads associated with each track are shown combined into a single unit. A number to be written in the store is first placed in the input register, and the number of the location to which it is to be transferred is placed in the address register. In addition to the information channels, the drum has one control channel on which synchronizing pulses are permanently recorded. The synchronizing pulses read from this track are passed into a counter with 10 binary stages. The number in

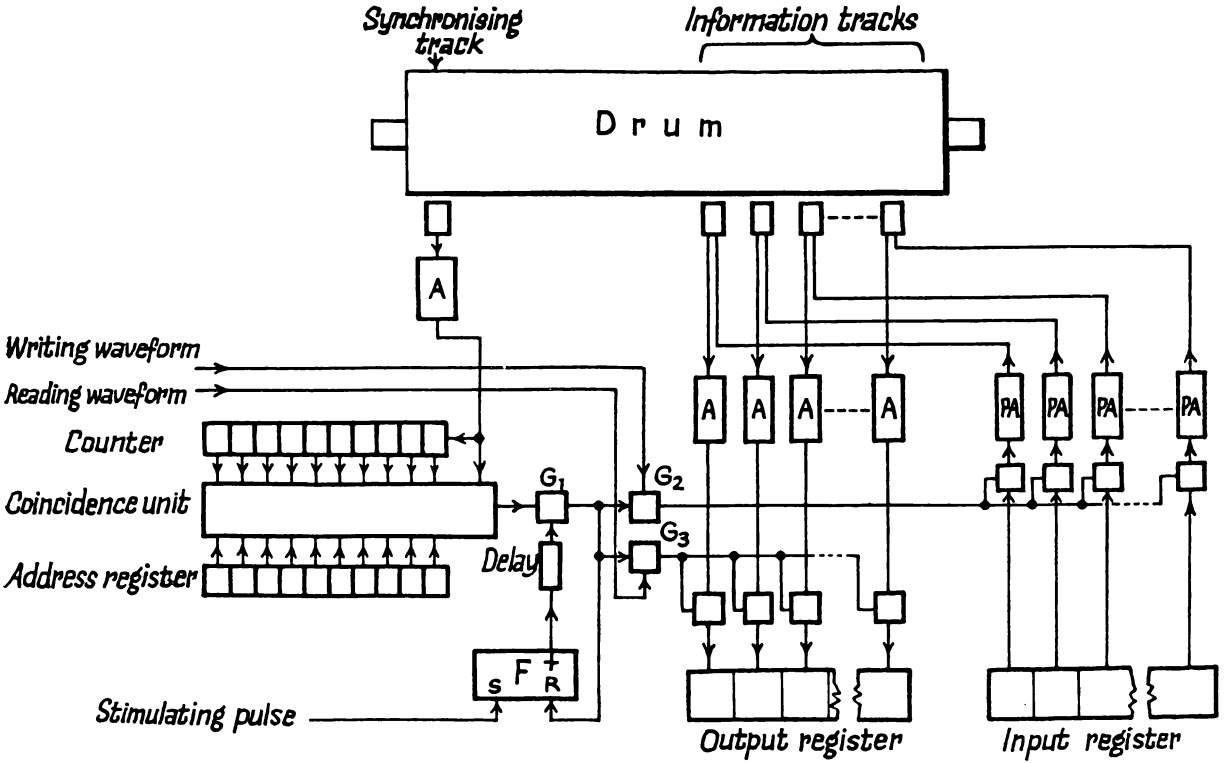


Fig. 5·20.—Logical design for a parallel store based on the use of a magnetic drum. A denotes an amplifier and pulse shaper; PA denotes a power amplifier.

this counter increases from 0 to 1,023 during each rotation of the drum and is passed in parallel form into a coincidence unit, where it is compared with the number standing in the address register. When these two numbers are the same a coincidence pulse is emitted. If the flip-flop F has previously been set by a stimulating pulse, and if the gate G_2 has been opened by the writing waveform, the coincidence pulse is applied to a set of gates through which information from the input register passes to the writing heads. At the same time the flip-flop F is reset. It is assumed that a stimulating pulse is always coincident in time with one of the synchronizing pulses, and the delay shown between F and G_1 (of duration half a pulse interval) is provided in order to ensure that the gate G_1 opens in advance of, and not during, the occurrence of a coincidence pulse. The action of reading is very similar to that of writing, except that G_3 is open instead of G_2 and the coincidence pulse is therefore routed so as to open a set of gates through which information can pass from the reading heads to the output register.

The serial store shown in Fig. 5·21 has a capacity for 256 words each of 32 binary digits. These are accommodated on eight tracks containing 32 words each. The same heads are used for reading and writing, and a relay tree is used for track selection. A relay is similarly used for changing over from reading to writing. In addition to the information channels there are three control channels. One of these has permanently recorded on it 1,024 synchronizing pulses. The second has 32 pulses equally spaced around it; these are known as minor cycle pulses and define the beginnings of successive word times. The third control track is known as the address track, and has permanently recorded in each word time one of the numbers from 0 to 31. The synchronizing pulses and the minor cycle pulses are passed to a waveform generator which produces four control waveforms W_1, W_2, W_3, W_4, whose functions will be described shortly.

The address register, in which is placed the number of the location to which access is required, is not shown in the diagram but is assumed to be of the circulating type (and synchronized with the drum), so that the number contained in it is available in serial form during each minor cycle. The outputs W_2, W_3, W_4, from the waveform generator, consist of pulses coincident in time with the pulses representing the three most significant digits of the number from the address register. These digits specify the track to which access is required, and the pulses representing them are isolated by means of three gates controlled by W_2, W_3, and W_4; the outputs of these gates are applied to three flip-flops which control the relays. The five least significant digits, which represent the position on the track of the location to which access is required, are similarly isolated by being passed through a gate controlled by W_1. The resulting pulses enter an anti-

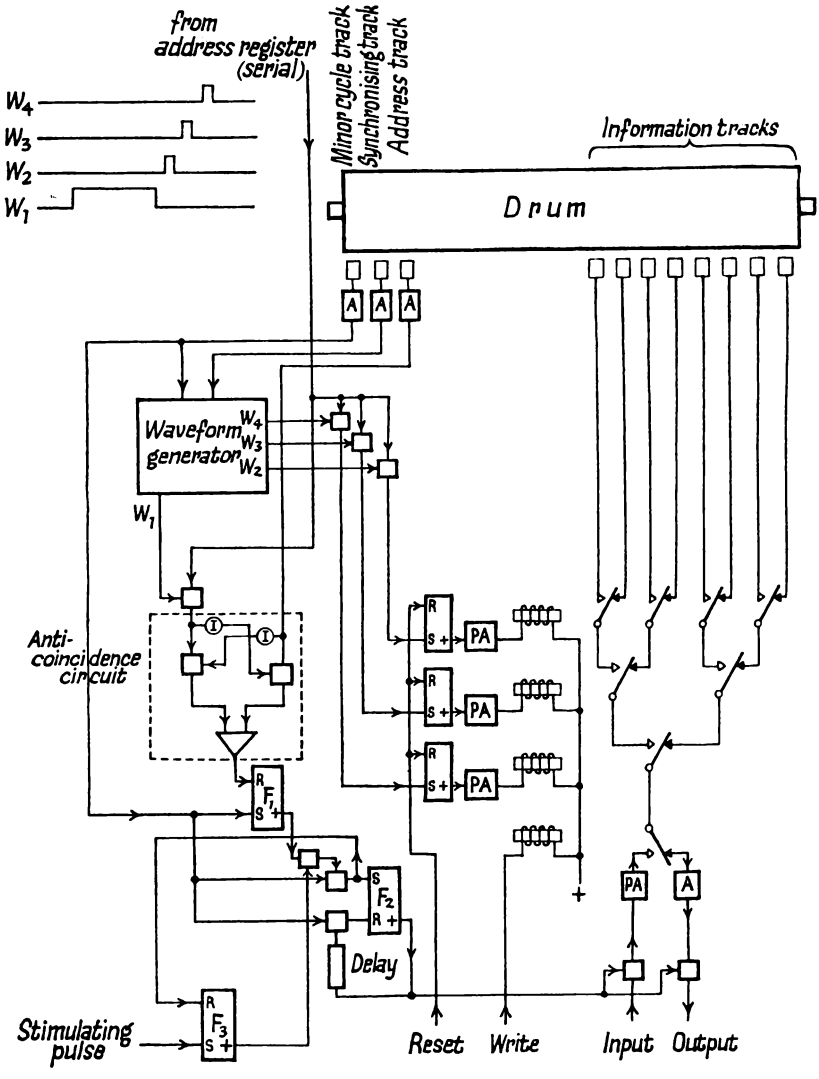


Fig. 5·21.—A serial store based on the use of a magnetic drum. A denotes an amplifier and pulse shaper; PA denotes a power amplifier.

coincidence circuit which is fed also with the pulses from the address track. It will easily be verified that no pulses are given out by this circuit during the minor cycle in which the number read from the address track is the same as the number coming from the address register; in

all other minor cycles one or more pulses are given out. The result is that the flip-flop F_1 (which is set at the beginning of each minor cycle) is reset before the end of each minor cycle with the exception of the one just mentioned. If F_1 has not been reset, and provided F_3 is set, then F_2 is set by the next minor cycle pulse and remains set for one complete minor cycle. The output of this flip-flop forms the coincidence waveform which opens the gates giving access to the head on the selected information track. The purpose of the flip-flop F_3 is to ensure that one coincidence waveform, and only one, is produced for each access. It will be seen that F_2 can never be set unless F_3 has first been set by the application of a stimulating pulse. Whenever the configuration of the relays is changed plenty of time must be allowed for them to settle down before the stimulating pulse is applied to F_3.

The use of an address track, with addresses permanently recorded on it, is an alternative to the use of a counter as in the store shown in Fig. 5·20. Which method is to be adopted for a particular store will depend partly on the personal preferences of the designer. The same may be said of many other decisions of a similar character, and the two schemes illustrated in Figs. 5·20 and 5·21 are to be regarded merely as examples chosen from a great number of possible schemes.

Magnetic Tape

Magnetic tape is usually made by spraying cellulose acetate, or paper, tape with a suspension of finely divided red or black oxide of iron, a suitable binder being included so that a comparatively tough layer of oxide about 0·0005 in. thick is formed. In the UNIVAC magnetic tape made of non-ferrous metal on which a layer of nickel has been deposited by plating is used.

A reel of plastic or paper tape 1,200 feet long can contain some 15,000 numbers with the equivalent of ten decimals, and has the advantage that it can be easily handled. Magnetic tape is, therefore, very suitable, both for input and output, and for auxiliary storage. If used alone for auxiliary storage there is the disadvantage of long access time, although in certain applications this is not a serious matter, particularly if several tape mechanisms are provided and provision is made for the machine to compute while hunting for information on the tapes. Magnetic tape can also be used advantageously to provide secondary auxiliary storage, of a semi-permanent character, in a machine in which primary auxiliary storage is provided by means of a drum. Many of the considerations which arise in the design of the reading and writing circuits are the same for tape as for drums; this is true also of certain aspects of logical design if a synchronizing track is provided on the tape. There is sufficient room on tape $\frac{1}{2}$ in. wide to provide about six channels on which information may be recorded independently.

Magnetic tape is usually run in contact with the heads, which may be similar to those used for sound recording. In such heads the core is built up in two separate parts from stampings, and the gap is formed by placing a non-magnetic shim between the parts before they are clamped together. As in the case of miniature heads, if care is not taken the stray field from the coil itself can cause trouble.

In principle, the fact that tape is run in contact with the head should enable digits to be packed more closely—for a given margin of operation—on tape than on a drum. However, this advantage cannot usually be realized because of the comparative non-uniformity of the coating of magnetic material on tape and because of the comparative crudeness of the mechanical arrangements; in practice, therefore, very similar recording densities are used. Magnetic tapes used in computers are usually run at much greater speeds than those used for sound recording. 100 in. per second (250 cm per second) is not an unusual figure; with 100 digits to the inch this gives a pulse interval of 100 μsec.

Even if they are worked with conservative recording densities, all systems using magnetic tape for input and output, or for auxiliary storage, suffer from the defect that digits can sometimes fail to be recorded, or fail to be properly read back, on account of isolated imperfections existing here and there in the tape itself, or of irregularities in the motion which cause the tape momentarily to lose contact with the head. Modern computing tape is relatively free from flaws, compared with the sound recording tape used in early experiments, and flapping of the tape on the heads can be greatly reduced by careful attention to mechanical design. It is, however, unwise to operate any system using magnetic tape without providing some degree of redundancy which can be used for checking or to reduce the rate of incidence of errors. One method is to record each digit on two separate channels with the heads spaced laterally along the tape. Since a flaw usually results in a signal being lost rather than in a spurious signal being introduced, it is found that quite good protection against errors due to flaws may be obtained by combining the outputs of the two channels through an 'or' circuit. This system is error-correcting rather than error-detecting, and will, subject to its essential limitations, enable a correct read-back to be obtained even if an error in recording has been made on one channel. If the degree of redundancy obtained by providing duplicate channels is thought to be excessive, error-detection, but not error-correction, may be obtained by recording an extra parity digit along with each number. An alternative method is to record a check sum after each block of numbers. This is a method which can be applied by the programmer even if no checking system has been built into the equipment.

If an error-detecting system is used, means should be provided

whereby the reading operation can be repeated when a check fails, since there is a good chance that after one or two attempts the information will be correctly read, especially if provision is made for the amplifier gain to be varied. The re-reading operation may require the manual intervention of an operator, but it is better that it should take place automatically. In the latter case the designer has the choice of providing special control equipment for the purpose, or of making it possible for the re-reading operation to be programmed.

An error-detecting system with automatic re-reading is, however, of no assistance if the information was recorded wrongly in the first case, and in some classes of work it would be desirable, given such a system, to read back information, for the purpose of checking, as soon as it has been recorded. This is time consuming, particularly if the tape can be read in one direction only, and there would seem here to be application for one of the error-correcting codes described in Chapter 6. Use, for example, of row and column parity digits as described on p. 252, with programmed error-correction, would enable a high degree of security against error to be obtained with much less redundancy of recording than in the two-channel system described above.

It is usual to provide some means of neutralizing permanent flaws on the tape. In systems in which synchronizing pulses are permanently recorded on the tape, and not written afresh each time a block of numbers is recorded, a simple method of neutralizing flaws is to omit the recording of synchronizing pulses in regions of the tape known to contain flaws. In the UNIVAC holes are punched in front of each flaw. A photo-electric cell senses the passage of each hole and causes the circuits to be rendered inoperative for a fixed time afterwards. Holes punched in the tape and sensed by photo-cells can be used for a variety of other control purposes, for example, for dividing the tape into blocks, in each of which a certain quantity of information can be stored.

The designer of an auxiliary storage system based on magnetic tape must decide to what extent the tape shall be controlled by specially built equipment and to what extent it shall be controlled by the programme. On the one hand, it is possible to design a system so that, when once the address (on the tape) of the particular piece of information required is delivered to a special register associated with the tape mechanism, the necessary hunting takes place automatically; in due course the required information is placed in a special set of registers, accessible to the machine, or in the high-speed store itself. On the other hand, it is possible to include in the order code of the machine orders which enable the tape to be started, reversed, and stopped in a manner called for by the programme and, similarly, to provide such other orders as are necessary to control the reading and writing

processes. Control of the tape can then be carried out by means of a subroutine. The advantage of this system is that it avoids the need to provide a good deal of special equipment which is used only when the tape mechanism is in motion. On the other hand, it will be found that the subroutine required for controlling the tape takes up an appreciable amount of space in the high-speed store. Furthermore, unless some special device is used, computing must be stopped during the time information is being hunted for on the tape.

Magnetic Wire

Steel piano wire, of diameter 0·004 in. or less, can be used successfully for recording digital information. A slightly better packing density, however, can be obtained by using special recording wire made by nickel-plating a non-ferrous wire. This wire has the further advantage that it is somewhat easier to handle than steel wire. A major difficulty encountered with all high-speed wire recorders, however, is to find a satisfactory method of repairing breaks in the wire. If it were not for this it is probable that wire would have been used more extensively for auxiliary storage purposes, since it enables a very large quantity of information to be stored in a small space. The problem is specially acute if ready interchangeability of spools of wire is required. Much of what has been said about the control of magnetic tape and the use of error-detecting systems applies also to magnetic wire, with the limitation that only one track is available for the recording of information.

The National Bureau of Standards have made an interesting application of wire recording on the SEAC. This is the use, for input and output, of small wire cartridges of the kind used in a commercial dictating machine. Each programmer is provided with one or more of these cartridges on which he may record programmes or input data (using punched paper tape as an intermediary), which can then be transferred into the machine at the rate of about 6,000 binary digits a second. The results, either final or intermediate, can be brought out in the same way.

STATIC MAGNETIC STORAGE

The systems described in this section have been made possible by the development of magnetic materials with high magnetic remanence, which have hysteresis loops of approximately rectangular form. Both magnetic alloys and ferrites with this property are now available. Such materials provide possibilities both for storage and for switching.

Fig. 5·22 represents a core, similar to a transformer core, made of one of the materials described above. It carries three windings *a*, *b*, and *c*, which for simplicity will be assumed all to have the same num-

ber of turns. The core may be used to store a single binary digit in the following manner. Suppose a pulse of current of sufficient intensity to saturate the core is passed in a given direction through winding *a*. The core then becomes magnetized in, say, the positive direction. A similar pulse passed in the same direction through winding *b* will cause only a slight temporary increase in the magnetization of the core. Consequently a detecting device connected to winding *c* will give only a small indication. Suppose now that the original pulse had been applied to winding *a* in the opposite direction, so that the core had become magnetized in a negative direction. The passage of a pulse in the original direction through *b* would then give a large output from winding *c*, since the direction of magnetization of the core would be changed.

We can regard the core as storing a 0 when magnetized in a positive direction and as storing a 1 when magnetized in the reverse direction.

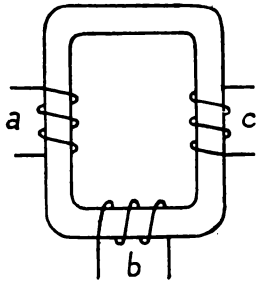


Fig. 5·22.

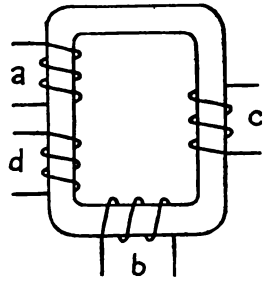


Fig. 5·23.

Writing consists in passing a sufficiently strong current in the appropriate direction through winding *a*; reading is performed by passing a current, always in the same direction, through winding *b* and examining the output from winding *c*. It will be observed that the act of reading destroys the information held in the core.

If another winding *d* is added, as shown in Fig. 5·23, the core can be used as a gate. A sufficiently large, steady current is passed through winding *b* to saturate the core strongly in the negative direction. We define a current of unit intensity as one which is just small enough, when passed through winding *a* or *d* in a positive direction, not to reverse the direction of magnetization of the core. A pulse of unit current applied to either *a* or *d* therefore results in only a small output from *c*. If, however, unit pulses are applied simultaneously to windings *a* and *d*, their combined effect is sufficient to reverse the direction of magnetization of the core, and a large output results.

Magnetic Hysteresis

The magnetic properties of a piece of material are usually displayed with the aid of a set of hysteresis loops similar to those for a typical rectangular loop ferrite shown in Fig. 5·24. Hysteresis loops are very convenient for discussing a situation in which the material is subjected to a periodically varying magnetizing force which reaches alternate positive and negative extreme values of equal moduli. Each loop then gives the variation of B with H for a particular amplitude of the exciting force. As the amplitude is increased the relevant loop expands until it coincides with the outer or major loop. Any further increase

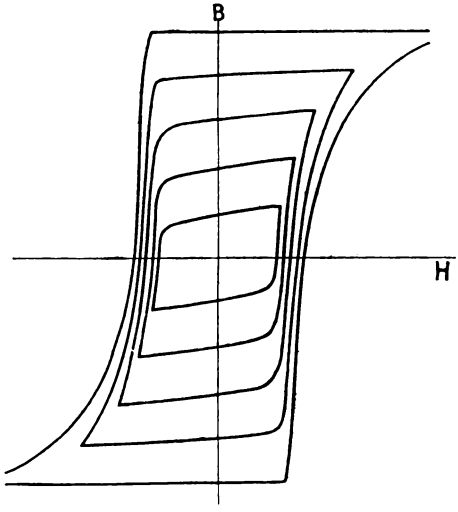


Fig. 5·24.—Hysteresis loops.

in the amplitude of H leads to little increase in the amplitude of B. The outer loop shown in Fig. 5·24 is not the true major loop, but, since a very large increase in the amplitude of H is needed to make it expand appreciably, it may, for practical purposes, be regarded as the major loop.

The curves of Fig. 5·24, however, give insufficient information for the discussion of situations in which the magnetizing force varies in a complicated manner, and those of Fig. 5·25 or their equivalent must be used instead. In this figure the continuous lines represent the dependence of B on H when H is increasing, and the dotted lines represent the dependence of B on H when H is decreasing. Thus if the initial magnetic state of the material is known, the variation of B as H is varied may be plotted by following initially a continuous or dotted

curve, according as H is increasing or decreasing, and changing over to the intersecting curve of the other type at points where the rate of change of H with time changes sign.

The action of the magnetic gating circuit described on p. 189 can now be explained in terms of the curves of Fig. 5·25. When no pulses are being applied, but unit current is flowing through the biasing winding, the core is in the magnetic state represented (say) by the point P. When a unit pulse of current is passed through either of the input windings the core is driven to the point Q, and a small change

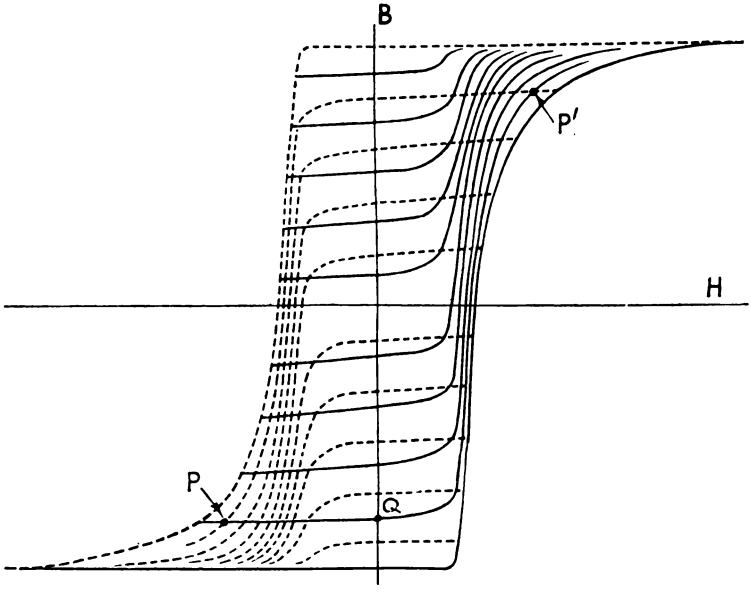


Fig. 5·25.—Magnetic hysteresis curves for a rectangular-loop ferrite.

in flux, represented by the vertical distance between Q and P, results. When unit pulses are applied to both input windings the core is driven to the point P' and there is a large change in flux, represented by the vertical distance between P' and P. In each case when the applied pulses have died away the core returns to P. In all cases the e.m.f. induced in the output winding is given by the rate of change of flux.

An approximate analysis of the action of the circuit may be obtained by using the idealized form of hysteresis loop shown in Fig. 5·26. Since, for each part of the curve, the induced e.m.f. is proportional to the rate of change of magnetizing force, the curve giving the induced e.m.f. as a function of time can be obtained by applying ordinary linear theory to each segment in turn, allowing for leakage

inductance and self- and mutual-capacity in the ordinary way. This kind of analysis is only possible in circuits in which the magnetic material is always returned to the same magnetic condition at the end of an operation.

The curves shown in Fig. 5·25 were obtained under static conditions. Any method of analysis based on them is, therefore, valid only if the rate of change of the magnetizing force is sufficiently small for it to be possible to regard the magnetic state of the material as being at all times in equilibrium with the applied field. This is usually expressed by saying that the applied field must not change substantially

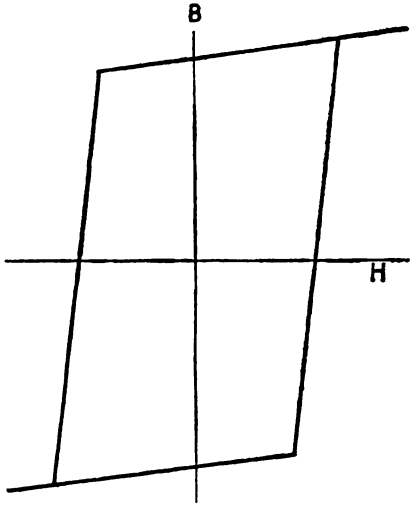


Fig. 5·26.—An idealized hysteresis loop.

during a period equal to the relaxation time of the magnetic material. The ferrite materials in use at the present time have a relaxation time of the order of 2 μsec. If H changes rapidly, the effect of the relaxation time of the material is to make B lag on H, and this lag, if not too great, can be allowed for empirically in applying curves of the type shown in Fig. 5·25.

Static Magnetic Delay Lines

The first application of magnetic cores with the properties described above was made by Aiken, Way Dong Woo, and An Wang in the Computation Laboratory of Harvard University. They constructed what they described as a static magnetic delay line, but which is better described as a shifting register. Its action depends on the fact that, if

the winding *c* of the core shown in Fig. 5·22 is connected to the winding *a* of another core, when the magnetization of the first core is reversed sufficient flux will be generated in the second core to reverse its magnetization also. In order that the effect of losses should be counteracted, it is desirable that the winding on the second core should have slightly more turns than the winding on the first core to which it is connected.

The circuit of the shifting register is shown in Fig. 5·27. It consists in effect of two separate registers, each containing as many cores as there are binary digits in the number to be shifted. The number is initially stored in the cores C_1, C_2, &c. When a pulse of current I_1 is passed through a winding on each of these cores, the number is transferred, by the action just described, to the second set of cores C_1',

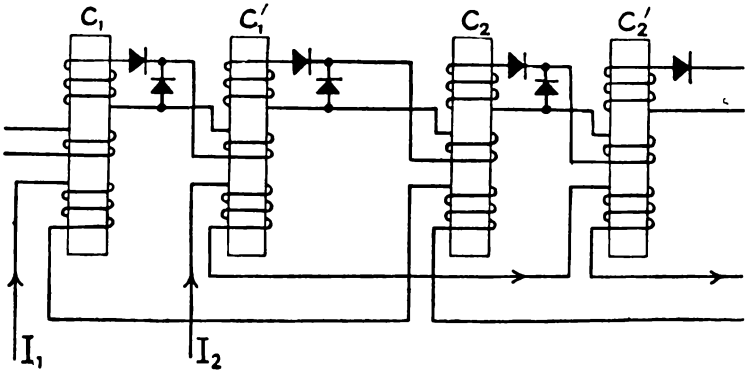


Fig. 5·27.—Magnetic shifting register or static magnetic delay line.

C_2', &c. If another pulse of current I_2 is then passed through a winding on each of this second set of cores, the number is transferred back to the first set, but shifted one digital position to the right. The purpose of the rectifiers, shown in the connecting links between the two sets of cores, is to prevent transmission of information in the wrong direction. If, instead of the single pulses I_1 and I_2, pulses are applied at regular intervals to the two shifting windings alternately, the set of cores acts like a delay line in that a number fed in at one end appears at the other end after a fixed interval of time, depending on the interval between the shifting pulses. If the output is connected to the input, a number once introduced circulates indefinitely. A delay unit formed in this way differs from an acoustic delay unit, however, in that the delay can be altered by altering the time interval between the shifting pulses and, if the shifting pulses cease to be applied, the number remains on the cores in static form.

Static magnetic delay lines with metallic cores, and working at a frequency of 16,000 shifts per second, have been used successfully in the Harvard Mark IV calculator. In this machine a magnetic drum is used for main storage, and the static magnetic delay lines, of which there are 200, are used for high-speed storage.

Coincident-current Magnetic Matrix Storage

If very small ferrite cores are made use of it is possible to achieve quite high speeds of operation. Moreover, the individual cores can be made sufficiently small for a high-speed store to be constructed with them, using one core per binary digit stored. The rectangular shape of the hysteresis loop enables an elegant solution to be obtained to the problem of switching to any required core. The method was originally proposed by J. W. Forrester, and the first full-sized store based on its use was constructed at M.I.T. and fitted to Whirlwind I in August 1953. This store could hold 1,024 words of 16 binary digits each; random access to any number could be obtained in 1 μsec, although a further 7 μsec had to elapse before a second reading or writing operation could be started. Writing into the store took 8 μsec. J. A. Rajchman was responsible for parallel development work on matrix-storage systems undertaken in the R.C.A. laboratories at Princeton, N.J.

The cores used in practice are only about 2 mm in external diameter, and the windings consist of single wires threaded through them. The cores are arranged to form a matrix in the manner shown in Fig. 5·28. If the hysteresis loop is nearly rectangular in shape it is possible to choose a value of current I such that, if current pulses of magnitude I are passed (in appropriate directions) through each of the wires threading a particular core, the direction of magnetization of that core is reversed, while, if a similar pulse is passed through one wire only, the core suffers no permanent change in direction of magnetization. This property enables the direction of magnetization of a selected core in the matrix to be reversed at will by passing current pulses of magnitude I simultaneously, in appropriate directions, through the x and y wires which intersect in that core; other cores threaded by the same wires receive only half excitation, and suffer no permanent change in their direction of magnetization. If a core is regarded as storing a 0 when magnetized in one direction and a 1 when magnetized in the other, the above procedure enables a 0 or a 1 to be written in any selected core.

The procedure used for reading is identical with that used for writing a 0, except that an output is taken from a winding which threads all the cores. If the selected core is already storing a 0, its direction of magnetization will remain unchanged and a small signal only will be induced in the reading winding; on the other hand, if the core is

storing a 1, its direction of magnetization will be reversed and a large signal will be induced. The signals obtained for 0s and 1s can be readily distinguished by means of a discriminating circuit. Information read in this way is destroyed at the same time, and it is normally necessary to make arrangements for it to be rewritten immediately after reading.

For a storage system based on the principles described above to be a practical possibility two requirements must be met. The first is that there should be no progressive degeneration of the information held in a core which is subjected to a large number of half excitations, or

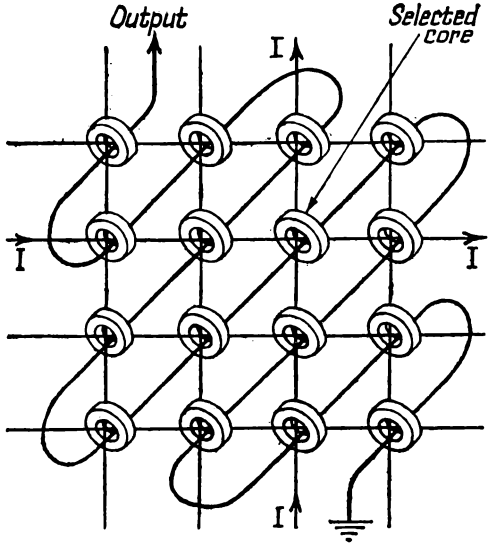


Fig. 5·28.—Matrix of storage cores.

'disturbances', without being selected for reading or writing. The second is that the sum of the unwanted signals arising from cores situated on the same x and y wires as the selected core (and which therefore receive half excitation during reading) should not be sufficiently large to make it difficult to discriminate between the wanted signals which are obtained when the selected core is storing a 0 or a 1 respectively. The first of these requirements concerns the properties of the material of which the cores are composed, and is met without difficulty by the ferrite materials now in use. It is found that the first disturbance produces an appreciable change, the second a very slight change, and that after some six to eight disturbances no further change is produced. More will be said about this matter in the section dealing with the testing of cores.

The second requirement is partly met in the original system designed at M.I.T. by threading the reading winding in opposite directions through alternate cores, so that the unwanted signals tend to cancel each other out. Threading the winding in this way means, however, that the wanted signal from a selected core can be of either polarity, and the amplifier must be designed accordingly. It is found that very similar outputs are obtained when a core which has had a 1 written in it is disturbed and when a core which has had a 0 written in it is disturbed—the ratio of the outputs is about 3 : 4—provided, in both cases, that the core has been disturbed at least once since the digit was written. In the original M.I.T. system the writing cycle included the application of a disturbing pulse.

The unwanted signal induced in the reading winding when a core is interrogated is the resultant of the signals from all those cores

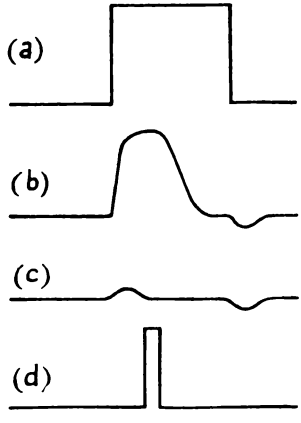


Fig. 5·29.—Waveforms illustrating the method of delayed sampling.

which lie either in the same row or in the same column as the selected core. Although this amplitude is much lower than it would be if the reading windings were threaded in the same direction through all the cores, it is still too large for reliable discrimination between 0s and 1s to be possible using a simple amplitude discriminator. In the M.I.T. system, therefore, use was also made of the fact that the pulse output from a given core is of longer duration when the polarity of the core is changed than when it is not changed. This may be made clear by reference to Fig. 5·29. The upper curve, (*a*), shows the waveform of a current pulse passed through the core and tending to magnetize it in, say, a positive direction. The lower curves show the waveform of the voltage induced in the reading winding when the core is initially magnetized in a negative direction (curve (*b*)), and when the core is

already magnetized in the positive direction (curve (*c*)). When the current pulse ends a small negative-going voltage is induced in each case. The duration of the early part of waveform (*c*) is determined by the time taken for the flux to build up and die away according to classical electro-magnetic theory; that of waveform (*b*) is of longer duration because the state of permanent magnetization of the material is being changed, and an explanation of the precise shape of the curve must be sought in theories of magnetic hysteresis in ferrites. In the M.I.T. storage system the reading winding is connected to the amplifier through a gate whose other input is fed with the waveform shown at (*d*) in Fig. 5·29. The signal is thus sampled at a time when the ratio of the amplitude of waveform (*b*) to that of waveform (*c*) is large, in practice about 200 : 1. This method will be referred to as the method of *delayed sampling*.

In order to reverse the magnetization of one of the cores used in the M.I.T. matrix store, it was necessary to pass about 1·6 amp through a single wire; thus I was about 0·8 amp. The circuit complications which would have arisen if such a current had to be passed in either direction through the same wire were avoided by providing two x wires and two y wires and connecting them to separate driving valves.

The store consisted altogether of 16 matrices, each with 32 rows and 32 columns. One digit of each word was stored in each matrix, so that, in all, storage was provided for 1,024 words each of 16 binary digits. In order to economize in driving valves the corresponding x and y wires threading each matrix were joined in series and connected to the same valve. This in no way complicated the reading operation, since, during reading, simultaneous interrogation of all the matrices is required. In order to make it possible to write 0s in some matrices and 1s in others, however, it was necessary to provide an *inhibiting* winding in each matrix. This winding threaded all the cores of the matrix and was similar to, though distinct from, the reading winding. Each inhibiting winding was connected to a separate driving valve. During the writing operation currents, both equal to I, were passed through the selected x and y windings. In matrices in which a 1 was to be written no current was passed through the inhibiting winding, but in those in which a 0 was to be written a current, also equal to I, was passed through the inhibiting winding in such a direction as to oppose the effect of the currents passed through the x and y windings. In this matrix, therefore, the selected core received a resultant excitation of half a unit, other cores on the same row or column as the selected core received zero excitation, and cores in other parts of the matrix received an excitation of half a unit in the negative sense. The total number of driving valves used was 128 for the x and y windings and 16 for the inhibiting windings, that is, 144 in all. Selection of the

x and y driving windings was achieved by means of conventional switching circuits using germanium diodes.

The use, in the pioneer system developed at M.I.T., of two x wires and two y wires, each with a D.C. connected driving valve, was dictated by a desire to use circuits which could be relied on to work with a minimum of development, so that attention might be concentrated on the solution of problems connected with the storage cores themselves. In later systems, however, some economy has been effected by using a single x wire and a single y wire fed from a pair of pulse transformers, each with a centre tapped primary winding. The transformers give a current step-up of about 4 : 1, so that it is possible to use two small valves for each co-ordinate instead of two large ones. Since the current waveform required for the reading and rewriting cycle is balanced, no difficulty arises with the use of pulse transformers, provided that they are properly designed. A balanced waveform is similarly required for writing in order that information previously written shall first be destroyed.

The foregoing remarks may be made clearer by reference to Fig. 5·30, which shows the waveforms in one possible system working on the principles described. Two sets are shown, corresponding respectively to the cases in which the digit stored in the selected core is 1 and 0. At (*a*) and (*b*) are shown the waveforms of the currents in the selected wires, and at (*c*) is shown the waveform of the e.m.f. which would be induced in the reading winding in the absence of any pulses in the inhibiting windings. If the waveform (*c*) is gated with the sampling pulse (*d*) and the result amplified and shaped, the result is as shown at (*e*); it will be seen that if the digit recorded is 1 a pulse is obtained, whereas if it is 0 no pulse is obtained. In the latter case, in order that the selected core may be left in the same condition as it was in initially, a pulse is passed through the inhibiting winding; this pulse is coincident with, and in opposition to, the negative-going portion of the current waveform in the x or y wire; the result is that the core receives a negative excitation of half a unit so that the 0 which has just been recorded is disturbed. A core that was originally storing a 1 will be automatically restored to its original condition if no current is passed through the inhibiting winding while current is passing through the x and y wires. In order that the rewritten 1 should be disturbed, however, it is necessary that a pulse should be passed through this winding when the rewriting operation is finished. The waveforms of the current which must be passed through the inhibiting winding in the two cases are shown at (*f*), and the resultant signal from the reading winding is shown at (*g*); it will be observed that the early parts of the latter waveform and of the waveform shown at (*c*) are the same, so that the gated and amplified signal output (*e*) is not affected by the passage of current through the inhibiting winding.

In the reading and rewriting cycle just described, the waveform (*f*) is derived directly from waveform (*e*) with the aid of a suitable switching circuit. For writing a similar cycle is used, but the waveform (*e*) is disregarded and the waveform (*f*) is made to depend on the digit being written into the store. During writing, the positive-going por-

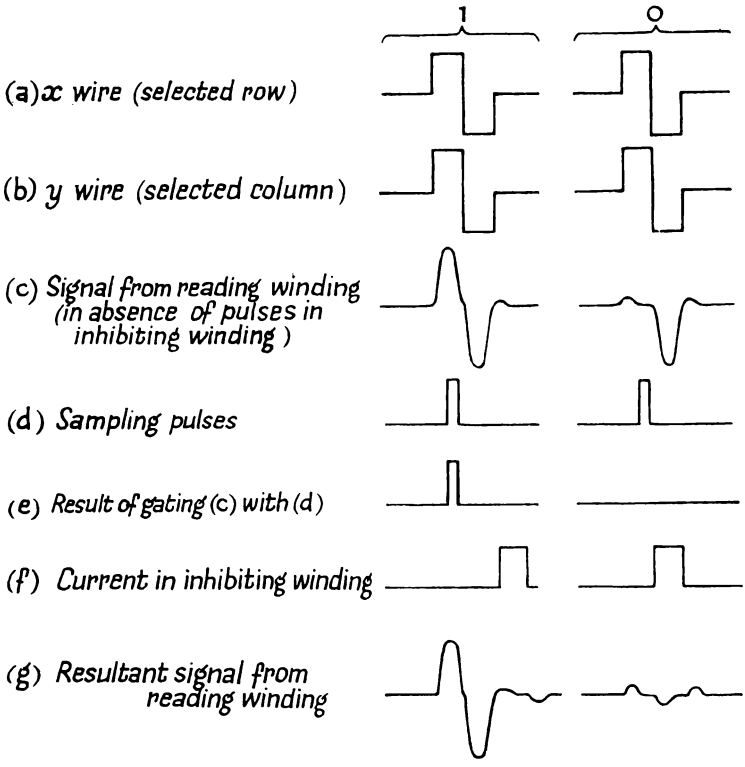

1
0
(a) x wire (selected row)
(b) y wire (selected column)
(c) Signal from reading winding (in absence of pulses in inhibiting winding)
(d) Sampling pulses
(e) Result of gating (c) with (d)
(f) Current in inhibiting winding
(g) Resultant signal from reading winding

Fig. 5·30.—Waveforms illustrating the operation of a coincident-current matrix store.

tions of the waveforms (*a*) and (*b*) serve the purpose of clearing information previously recorded.

It will be observed that the current in the inhibiting winding is not balanced about zero, so that the system as described does not lend itself to the use of a pulse transformer for driving this winding. This is perhaps no great disadvantage, since it is necessary for the current to be passed in one direction only, but if desired the waveform can be made balanced by following each positive-going pulse with an equal negative-going pulse. This merely acts as another disturbing pulse

and does no harm; it does, however, result in a slight lengthening of the operating cycle.

Alternative Coincident-current Storage Systems

The primary requirements which cores must satisfy, in order to lend themselves to the construction of a storage system of the type so far discussed, are that the responses obtained by interrogating cores in which 0s and 1s are stored should be sufficiently distinct and that the information should not be destroyed by any sequence, however long, of disturbing pulses. In order, however, that unwanted signals should be sufficiently reduced by cancellation, it is necessary that certain secondary requirements should also be met. For example, the unwanted signal resulting from disturbing a core containing a 0 should be as near as possible to that obtained by disturbing a 1, and a single disturbance applied as part of the writing cycle should be sufficient to bring the core sensibly to its asymptotic condition. It is not found possible to meet both primary and secondary requirements by close control of the manufacturing processes alone, and resort must be had to grading of the output of cores, only those with responses falling between closely defined limits being used in the construction of a particular matrix.

The requirements as regards cancellation of unwanted signals can be eased by applying the reading pulse to the x wire (say) in advance of the pulse applied to the y wire. Both pulses terminate at the same instant, the x pulse being, therefore, longer than the y pulse. The effect of this is that the unwanted signals, produced by disturbing the non-selected cores on the selected row, have time to die substantially to zero before the selected core is switched. In a matrix with 32 rows and 32 columns, therefore, only 31 cores contribute to the unwanted signal instead of 62. A further improvement can be effected, at the cost of a slight increase in the number of driving valves and associated decoding circuits, by making the matrix rectangular instead of square; for example, if the matrix has 64 rows and 16 columns, instead of 32 rows and 32 columns, the number of driving circuits is increased from 64 to 80, but the number of cores contributing to the unwanted signal is reduced from 31 to 15.

In the system just described the reading winding is threaded in opposite directions through alternate cores so that unwanted contributions to the total signal tend to cancel; further, delayed sampling of the resultant signal is used. An alternative method of neutralizing the effects of unwanted signals from disturbed cores is to integrate the signal from the reading winding, which may then be threaded in the same direction through all cores. This method depends on the fact that when a core is disturbed it is returned very nearly to its original magnetic condition. The integrated e.m.f. induced in the reading

winding, being proportional to the integrated rate of change of flux, is therefore nearly zero. A core which is switched, on the other hand, does not return to its original magnetic condition, and the integrated signal is not zero. It will be noted that the effectiveness of this method does not depend on the cores giving identical outputs, although it does require that a core shall reach very nearly its asymptotic condition after a single disturbance. Its major disadvantage is that integrating circuits of the necessary quality are rather complicated, and one such circuit is required for each digit in the word. This disadvantage is less serious in the case of very large stores, and it is to matrices with 10,000 or more elements that the system is likely to have application. Rajchman has demonstrated experimentally a single matrix of this size.

The coincident-current storage systems so far described depend on the ability of the cores to discriminate between currents having the ratio 2:1, the stronger current being sufficient to saturate the core, and the weaker current being insufficient to produce any permanent effect. This requirement is eased if, in addition to passing a current $I/2$ through the selected x and y wires of the matrix, a current $-I/6$ is passed simultaneously through all the other wires. It may easily be verified that the selected core will then experience a magnetizing force equivalent to a current I and that no other core will experience a magnetizing force equivalent to a current of greater magnitude than $I/3$. The cores are thus called upon to discriminate between currents in the ratio 3 : 1 instead of 2 : 1 as formerly. An equivalent system, which may lead in practice to simpler switching arrangements, is to have an extra winding threading each core; a current $2I/3$ is then passed through each selected x and y wire and a current $-I/3$ through the new winding. These currents produce the same excitations in the various cores as in the previous case. By using more windings and more complicated switching arrangements still more favourable current ratios may be obtained.

When they were first devised multiple-coincidence systems of the type just described, particularly the one which gives a 3 : 1 discriminating ratio, appeared to represent a distinct advance in the logical design of matrix storage systems. They suffer, however, from the serious disadvantage that all the cores in the matrix (other than the selected core) are disturbed, and hence contribute to the unwanted signal, instead of only those cores lying on the same row or column as the selected core. Moreover, some cores are disturbed positively and some negatively, so that effective cancellation of unwanted signals is impossible. With the cores available at the present time these disadvantages more than offset the advantage of the unwanted signal from an individual core being smaller.

Decoding by the Use of Magnetic Cores

By the use of the matrix arrangement the problem of selecting a particular core is reduced to the selection of one x wire and one y wire. This may be done, as in the original M.I.T. system, by means of conventional electronic switching circuits. It is possible, however, to make use of magnetic cores for this purpose also and, while this may be hardly worth while in the case of a small matrix, it may effect considerable economy in the case of a very large matrix.

Fig. 5·31 shows one method by which a decoding tree, which

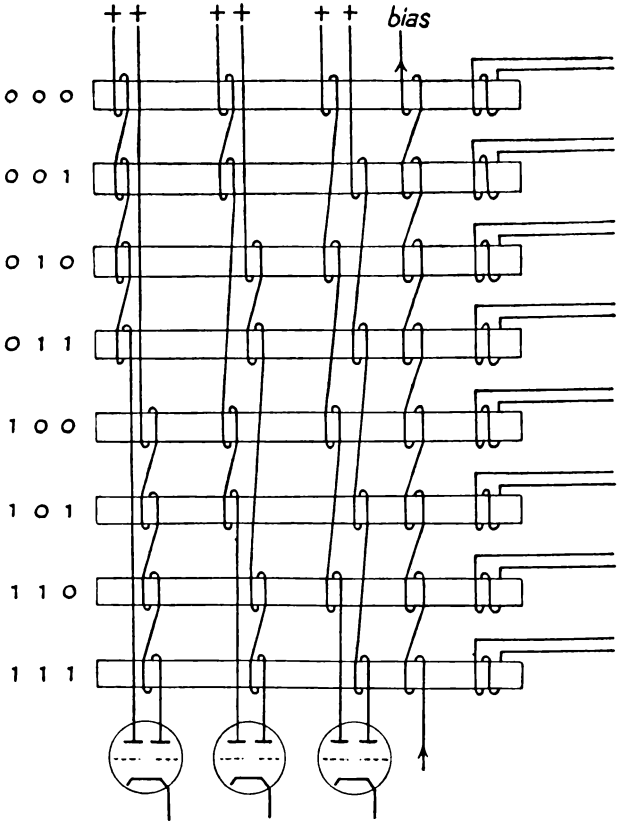


Fig. 5·31.—A magnetic decoding tree.

accepts three binary inputs and gives a current pulse in one of eight outputs, may be constructed with the aid of magnetic cores. There are eight cores, one corresponding to each output, and each core car-

ries one winding—a digit winding—for each binary digit, and a biasing winding through which a steady current is passed. At the moment of switching, a current pulse is passed through each digit winding, in one direction if the binary input concerned carries a 1, and in the other direction if it carries a 0. The strength of the biasing current is chosen so that only that core for which the currents passed through the digit windings augment each other is switched. This core, therefore, gives a large output, whereas the other cores give a small output only. It will be seen that the decoding tree just described consists essentially of eight three-input gates of the type described on p. 189.

It should be noted that, when magnetic cores are used for switching, a high operating speed can be obtained by using current pulses of large amplitude. The situation is quite different in this respect from that in a coincident-current storage matrix, where the switching current I is limited by the fact that a current $I/2$ must be insufficient to switch the matrix. If the output winding of a switched core is suitably loaded an appreciable amount of power may be obtained from it, provided, of course, that sufficient power is supplied to the input winding. The action of the core in this respect resembles that of an ordinary transformer.

Decoding trees of the kind just described may be constructed with a larger number of binary inputs. However, since the total number of turns which have to be wound on each core increases with the number of binary inputs, the system is practicable only when the number of binary inputs is quite small. Switches of this type, however, might be used to drive directly a storage matrix with 16 or even 32 rows and columns.

When a decoding tree with a large number of outputs is required, use may be made of a matrix of cores similar to that shown in Fig. 5·32. This bears a superficial resemblance to a coincident-current storage matrix, but its mode of operation is somewhat different. The cores used are rather larger, and each carries three windings of 20 turns or more. Two windings on each core are connected to form part of an x wire and a y wire of the matrix, and the third delivers the output pulse. When the matrix is quiescent a steady current of at least twice the intensity required to saturate a core passes through each y wire and no current passes through the x wires. When the matrix is operated the current flowing through one of the y wires is interrupted, and a current pulse of sufficient strength to saturate a core is passed through one of the x wires. The result is that the direction of magnetization of the core lying on the intersection of the selected x and y wires is reversed and an e.m.f. is induced in its output winding. Other cores lying on the selected x and y wires do not have their directions of magnetization reversed, although the magnetizing force falls to

zero and a small unwanted output is produced. This unwanted output can be minimized by arranging that the current in the selected y wire should fall to zero slowly and in advance of the application of a pulse to the x wire; similarly, the rise of current in the y wire may take place slowly, after the current in the x wire has fallen to zero. There is a further pulse in the same direction which occurs when the current through the selected y wire is restored; if this restoration takes place slowly the pulse will be of small amplitude, but its area is such

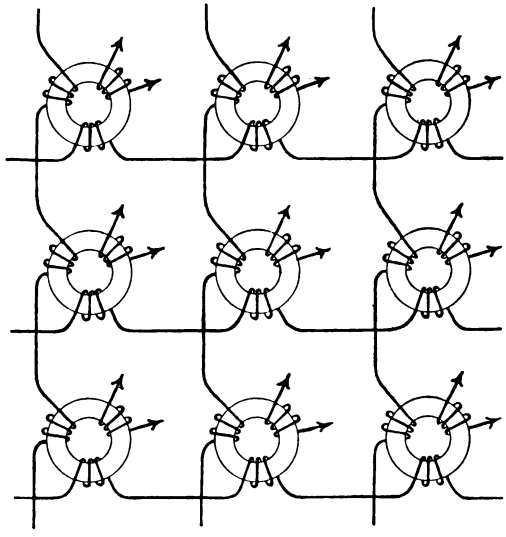


Fig. 5·32.—A matrix switch.

as to make the whole waveform of the e.m.f. induced in the output winding balanced about zero.

An alternative system is to make use of a bias winding which threads all the cores and carries a steady current sufficient to saturate the cores in one direction. Selection is then done by applying similar pulses of current to a pair of x and y wires. This arrangement has the advantage of saving power in the driving valves, but presents the designer with the problem of providing an accurately stabilized steady current for the bias winding.

Each time the direction of magnetization of a core is changed a quantity of electrical energy, proportional to the area enclosed by the hysteresis loop, is converted to heat. If the temperature of the material approaches the Curie point—about 300° C in the case of ferrites—its magnetic properties become modified, and at the Curie point itself all ferro-magnetic properties disappear. Thus, heating of the

material sets a limit to the number of times a second a magnetic core may be switched. Heating is not of practical importance in the case of the very small cores used for storage, but must be taken into account when designing a switching system using larger cores. In the case of switching cores of 8 mm in diameter the limit is in the neigh-

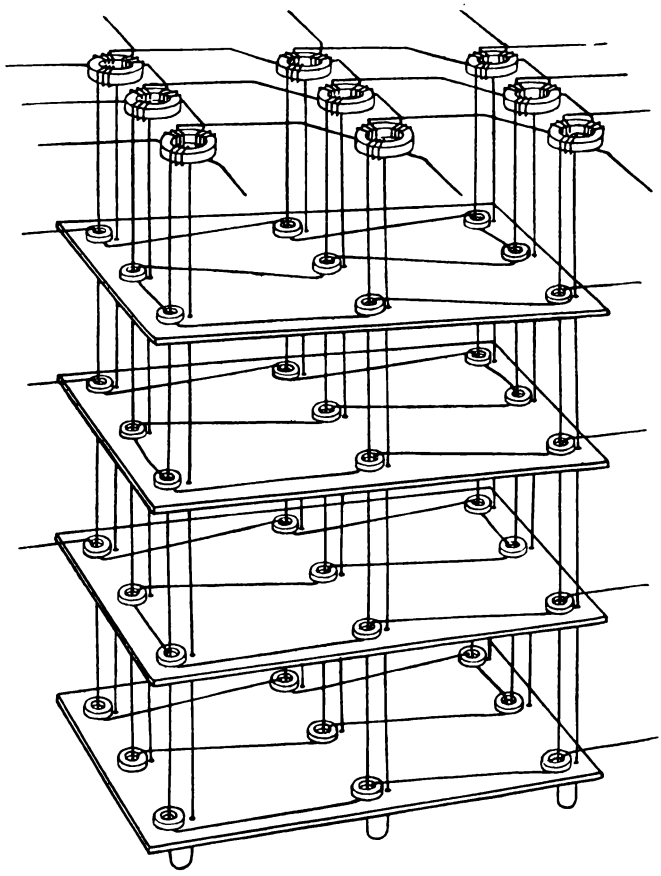


Fig. 5·33.—A storage system using a matrix switch.

bourhood of 10^5 reversals per second, although this may be extended if special measures are taken for cooling the cores. If the limit is exceeded the core loses its switching properties, but these return as soon as it cools.

If a switching matrix is available, with a number of outputs equal to the number of words to be stored, the form of matrix store illustrated in Fig. 5·33 can be used. In this store the coincident-current

principle is used for writing but not for reading, so that the problem of distinguishing between the wanted signal from the selected core and unwanted signals from disturbed cores is avoided.

Fig. 5·33 shows for simplicity a store having capacity for nine words of four digits each. The cores which store the digits of a particular word are all arranged on the same vertical wire, and each vertical wire is connected to one of the outputs of the switching matrix. The cores in any one horizontal plane are threaded by a wire which will be referred to as the digit winding.

It will be supposed that initially some of the storage cores are magnetized in one direction and some in the other, according as they are storing 0s or 1s. To perform a reading operation a pulse of current is caused to flow through one of the vertical wires by selecting appropriate x and y lines of the switching matrix; a large e.m.f. is induced in the digit windings which thread cores whose magnetization is reversed by the passage of the current, while a small e.m.f. only is induced in the other digit windings. If the switching matrix were perfect in its action current would flow in one vertical wire only, so that one core, and one alone, would provide the e.m.f. induced in each digit winding; in practice, owing to imperfect action of the switching matrix, currents will flow in other vertical wires, but these will be very small and will give rise to entirely negligible outputs from the storage cores through which they flow.

The reading operation just described is always followed by a writing operation, and for this purpose use is made of the negative-going output pulse from the switching matrix which follows the initial positive-going pulse. This negative-going pulse tends to write 1s in all the cores lying on the selected vertical wire; it is prevented from doing so, in those cores in which a 0 is to be written, by the passage of an inhibiting current through the digit winding. The current also passes through all the other cores in the same horizontal plane, and must be weak enough not to switch these cores but at the same time strong enough to prevent the switching of the selected core. The same considerations are here involved as in the original coincident-current switching system described earlier, and the conditions can be met, if the amplitude I of the writing pulse is suitably chosen, by making the inhibiting pulse of amplitude $I/2$.

Manufacture of Magnetic Cores

The principal constituents of the material from which ferrite cores are made are magnesium oxide (MgO), manganese oxide (MnO), and ferric oxide (Fe_2O_3). These materials, in a finely ground form, are mixed with a suitable binder and moulded under pressure to form cores of the required shape and size. These are then fired under closely controlled conditions in a furnace whose temperature can be

maintained constant to within a few degrees. A considerable amount of shrinkage takes place during the firing process, and must be allowed for when deciding on the size of the mould.

Magnetic cores made from magnetic alloys are usually formed by winding a thin strip of the alloy around a bobbin. Such cores are much more expensive than ferrite cores, and even if strip as thin as 0·000125 in. is used they cannot be made as fast in operation. It seems that for most purposes ferrite cores will displace metallic cores altogether, though the latter have one advantage which may lead to their continued use in certain switching applications. This is that the slope of the hysteresis loop for large H is smaller than for ferrite cores, so that switches may be designed which give a smaller unwanted breakthrough signal.

The reason why cores of a certain composition have approximately rectangular hysteresis loops is not known. A similar property can be induced in some materials which do not normally possess it by subjecting them to strain, and it has been suggested that residual strains in the material account for the observed rectangularity of the hysteresis loops of some ferrites. The same effect is, however, observed in mouldings of many different shapes and sizes, so that there would appear to be some difficulty in accepting this explanation.

The coercive force of a material, and hence the amount of current required for switching, depends on its composition. Small storage cores, with about one-third the coercivity of those whose characteristics have been given earlier, have been produced. Unfortunately, however, it is found that these cores are much slower in operation, requiring under comparable conditions about five times as long for a reversal of magnetization. The reason for this apparent connexion between coercive force and speed of switching is not understood, but it is interesting to note that a similar connexion is found in the case of laminated metallic cores.

Testing of Ferrite Cores

It is clearly necessary that every core used in a store should come up to a certain minimum standard of performance. Since a store may contain anything from twenty thousand cores upwards, and since the replacement of a faulty core, once it has been wired into a matrix, is a troublesome operation, there is a need for a testing procedure which can be applied to a large number of cores. This procedure may then be used both to reject cores which are wholly unsatisfactory and to grade the remainder according to prescribed characteristics. Experience has shown that it is best to use a method of test which simulates as closely as possible the conditions which obtain in the storage system it is intended to use; the examination of hysteresis loops displayed

on a cathode-ray tube screen does not constitute an adequate testing procedure.

In one of the testing procedures used in selecting cores for the original M.I.T. storage system, a current having a waveform similar to that shown in Fig. 5·34 was passed through a single wire threaded through the core, and the signal induced in another similar wire was displayed on an oscilloscope. The waveform falls into two parts (*a*) and (*b*). (*a*) simulates the writing of a 1 into a core, the disturbing of the core by a pulse of half amplitude applied in the opposite direction, and the reading of the stored 1 by a full-amplitude pulse in the same direction as the disturbing pulse. (*b*) is similar to (*a*), except that the writing pulse is followed by eight disturbing pulses, so that, after having the 1 written in it, the core is disturbed eight times before the reading pulse is applied. In order to enable a direct comparison to be

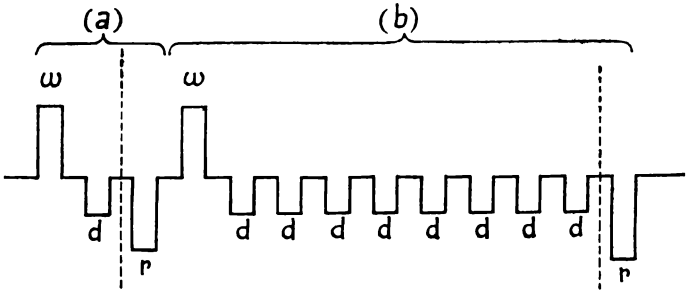


Fig. 5·34.—Current waveform used for testing storage cores. Writing pulses are indicated by *w*, disturbing pulses by *d*, and reading pulses by *r*.

made of the signals obtained from a 1 which has been disturbed once and from a 1 which has been disturbed eight times, an oscilloscope timebase is triggered at the times shown by the vertical dotted lines so that the relevant waveforms are superposed. If a single disturbance is sufficient to bring the core to its asymptotic condition the two waveforms will coincide. The test can be varied by modifying the current waveform so that the result of reading a 1 from a core which has been disturbed *n* times is compared with the result of reading a 1 from a core which has been disturbed *m* times, where *n* and *m* are arbitrarily chosen integers.

When testing cores for use in the system described on pp. 205–6 it is convenient to be able to adjust independently the amplitudes of the writing, reading, and disturbing pulses. When a new batch of cores is to be graded, samples must be taken and measurements performed on them with a view to deciding on the optimum values for the reading and writing currents. If the current is too large a core may not reach

its asymptotic condition until it has been disturbed a considerable number of times; on the other hand, if the current is too small correct switching action will not be obtained. Again there is the desirability of securing an optimum ratio between the signals obtained by reading a 1 and by reading a 0. If the cores are to be used in the 'delayed sampling' system, then the ratio of the signals must be determined at a certain time τ after the application of the reading pulse, and the optimum value of τ must also be decided on. This test is difficult to perform with a single core on account of the small response obtained when reading a 0, and it may be necessary to use 50 or more cores all threaded by the same wires. Once the optimum value for the reading and writing current has been decided on, each core must be tested to see that its performance falls within specified limits; since a large number of cores have to be tested, some simple and rapid way of doing this is necessary. Various automatic devices have been constructed which enable cores to be graded at a considerable speed.

The stringency of the specification the cores must meet depends entirely on the system in which the cores are to be used. A system depending on cancellation for the elimination of unwanted signals calls for cores of a very high degree of uniformity, whereas a system in which the selected core alone contributes appreciably to the total signal requires only that the peak amplitude of the pulse obtained by reading a 1 shall be greater than a certain value, and that the peak amplitude of the pulse obtained by reading a 0 shall be less than some other value.

CHAPTER 6

ELECTRONIC SWITCHING AND COMPUTING CIRCUITS

THE design of switching and control circuits for use in an electronic computing machine has already been touched on in earlier chapters; a more detailed and systematic discussion of the subject will now be given.

SWITCHING ALGEBRA

Some assistance in the understanding of complicated switching systems can be obtained from the use of a symbolic notation. Various forms of symbolism based on the use of Boolean algebra have been applied for this purpose. They all stem from a paper by C. E. Shannon in which Boolean algebra was first applied to the discussion of relay circuits. The Computation Laboratory of Harvard University have put forward a closely related system in which the manipulation is carried out according to the rules of ordinary algebra.

Consider a simple gate, as shown in Fig. 6·1, with signals x and y applied to its inputs, and suppose that the output is w. x, y, and w

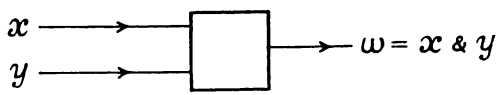


Fig. 6·1.—An 'and' circuit, or gate.

all represent binary quantities, and can therefore have the values 0 and 1 only. The following table gives w as a function of x and y:

x	y	w
0	0	0
0	1	0
1	0	0
1	1	1

It is convenient to have a piece of symbolism to express the relation between w, x, and y given by the above table, and we write

$$w = x \,\&\, y.$$

Note that w is a symmetric function of x and y so that

$$x \,\&\, y = y \,\&\, x.$$

If there are three variables, x, y, and z, it is easy to verify by considering all possible cases that

$$(x \,\&\, y) \,\&\, z = x \,\&\, (y \,\&\, z),$$

so that there is no ambiguity in writing

$$x \,\&\, y \,\&\, z,$$

and, moreover, the terms in this expression may be permuted. The three-input gate, shown in Fig. 6·2, produces $x \,\&\, y \,\&\, z$ directly when fed with x, y, and z.

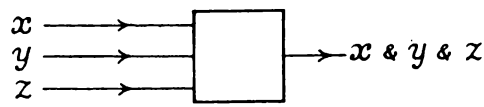


Fig. 6·2.—A three-input 'and' circuit.

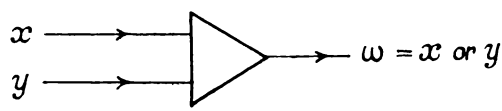


Fig. 6·3.—An 'or' circuit.

Similarly, the action of the 'or' circuit shown in Fig. 6·3 is given by the table

x	y	w
0	0	0
1	0	1
0	1	1
1	1	1

and may be expressed by the symbolic notation

$$w = x \text{ or } y.*$$

Here again the terms may be permuted, and it may be verified that there is no ambiguity in writing

$$x \text{ or } y \text{ or } z.$$

The three-input 'or' circuit, shown in Fig. 6·4, gives this output directly when fed with x, y, and z.

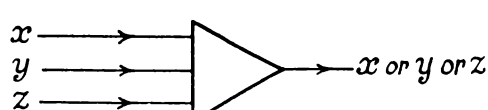


Fig. 6·4.—A three-input 'or' circuit.

* This is the 'inclusive or'; the 'exclusive or' is the sum (modulo 2) of x and y denoted later by $x \oplus y$.

The action of the inverting circuit shown in Fig. 6·5 is given by the table

x	w
0	1
1	0

and we may write

$$w = not\ x,$$

or alternatively

$$w = x'.$$

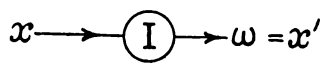


Fig. 6·5.—An inverting circuit.

The symbolism just explained is sufficient to describe the action of any digital switching circuit built up out of the elements referred to.

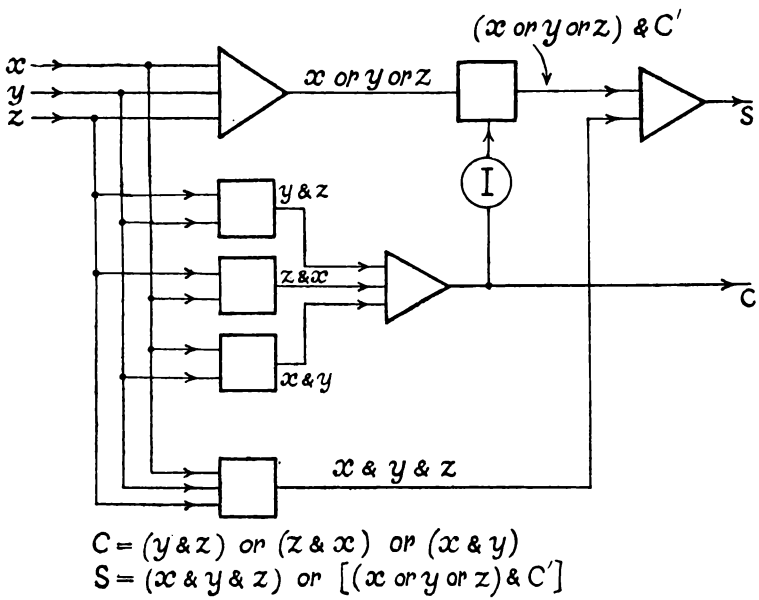


Fig. 6·6.—The adding circuit of Fig. 1·21 with annotations in Boolean algebra.

As an example, the adding circuit illustrated in Fig. 1·21 and described in Chapter 1 is reproduced in Fig. 6·6 with annotations explaining its action in symbolic language.

The symbolism given above is derived from the subject of symbolic logic. This is possible, since a formal analogy exists between symbolic logic and the theory of switching circuits. In symbolic logic the following interpretation would be placed on the values taken by the variable x:

$x=1$, the proposition x is true;
$x=0$, the proposition x is false.

x & y means the proposition which is true when both x and y are true, and false otherwise; x *or* y means the proposition which is true when either x or y, or both, are true, and false otherwise; and *not* x is the proposition which is true when x is false, and vice versa. The rules by which statements about propositions expressed in terms of this symbolism should be manipulated were discussed by G. Boole in 1847, and the subject is known as Boolean algebra by analogy with ordinary algebra. Ordinary algebra is usually regarded as being a generalized form of arithmetic, but it can also be regarded as a subject in which symbols are manipulated according to a set of axioms or, to express it less formally, as a game played according to certain rules. From this point of view, Boolean algebra is another game in which the rules differ from those of ordinary algebra and in which the symbols do not carry arithmetic meanings. The analogy with ordinary algebra, however, suggests using the ordinary algebraic signs for addition and multiplication to represent the logical operations of '*or*' and '&'; thus we write

$x+y$ for x *or* y,
and xy for x & y.

After a little experience it will be found that confusion rarely arises from the use of the signs for addition and multiplication in this way. However, if doubt is felt about this, the symbols '&' and 'or' may be retained. If a shorter expression is desired for 'or' the symbol v may be used in its place. $\sim x$ is sometimes used for *not* x. The postulates of Boolean algebra in which the variables may take only the values 0 and 1 can be written as follows:

$0+0=0$	$0.0=0$
$0+1=1$	$0.1=0$
$1+0=1$	$1.0=0$
$1+1=1$	$1.1=1$

Various theorems of Boolean algebra may now be proved. Since each variable can take only two values, the simplest mode of proof in

all cases is a straightforward verification. The first five theorems are true also in ordinary algebra.

$$x+y=y+x \tag{1}$$
$$xy=yx \tag{2}$$
$$x+(y+z)=(x+y)+z \quad (=x+y+z) \tag{3}$$
$$(xy)z=x(yz) \quad (=xyz) \tag{4}$$
$$x(y+z)=xy+xz \tag{5}$$
$$x+x=x \tag{6}$$
$$xx=x \tag{7}$$
$$1+x=1 \tag{8}$$
$$x+x'y=x+y \tag{9}$$
$$(x')'=x \tag{10}$$
$$(x+y)'=x'y', \qquad (x+y+z)'=x'y'z' \ * \tag{11}$$
$$(xy)'=x'+y', \qquad (xyz)'=x'+y'+z' \ * \tag{12}$$
$$x+x'=1 \tag{13}$$
$$xx'=0 \tag{14}$$

It should be particularly noted that no meaning is given to the minus sign in Boolean algebra.

Examination of the postulates of Boolean algebra given above show that they are unchanged if 0s are replaced by 1s and plus signs by multiplication signs, and vice versa. From any theorem a dual theorem can, therefore, be inferred without further proof. Several theorems in the above list are duals of other theorems also on the list; for example, (1) and (2), (11) and (12) are duals of each other. In applications of Boolean algebra a choice may be made between two completely parallel systems, one being the dual of the other. For example, in his original paper, Shannon pointed out that when discussing relay circuits it is possible either to take 0 to represent a closed circuit and 1 to represent an open circuit, or vice versa. There does not seem to be a great deal to choose between the two alternatives, and the former has been standardized for use in the Bell Telephone Laboratories. In this notation the circuit formed by two relay contacts, x and y, connected in series, is represented by $x+y$ and that formed by two contacts connected in parallel by xy. Sometimes it is useful to consider the dual of a given relay circuit; for example, the dual of two normally open relay contacts in series, represented by $x+y$, is the circuit represented by $(x+y)'$, which by (11) equals $x'y'$, that is, the dual circuit consists of two normally closed contacts in parallel.

It is sometimes convenient to have a notation for the sum, modulo 2, of x and y, and we may write this as $x \oplus y$. Note that

$$x \oplus y = xy' + x'y,$$

* May be extended to any number of terms.

where the right-hand side is an expression in Boolean algebra. The sum output of a three-input binary adder may be written

$$\begin{aligned} S(x, y, z) &= x \oplus y \oplus z \\ &= (xy' + x'y)z' + (xy' + x'y)'z \\ &= (xy' + x'y)z' + (xy')'(x'y)'z && \text{by (11)} \\ &= (xy' + x'y)z' + (x' + y)(x + y')z && \text{by (12)} \\ &= xy'z' + x'yz' + x'y'z + xyz. && \text{by (14)} \end{aligned}$$

Inspection of this result shows that it correctly represents the sum digit. The sum output of the adder shown in Fig. 6·6 can be transformed into this form by the application of certain of the theorems given above. We have

$$\begin{aligned} S(x, y, z) &= (x + y + z)(xy + yz + zx)' + xyz && (15) \\ &= (x + y + z)(xy)'(yz)'(zx)' + xyz && \text{by (11)} \\ &= (x + y + z)(x' + y')(y' + z')(z' + x') + xyz && \text{by (12)} \\ &= xy'z' + x'yz' + x'y'z + xyz, && \text{by (7) and (14)} \end{aligned}$$

which is the same as the expression obtained above. The carry output of the adder is expressed as follows:

$$C(x, y, z) = yz + zx + xy.$$

This is clearly a correct representation of the carry digit since it vanishes unless at least two of x, y, z are equal to 1.

It is very easy to express the action of a given circuit in terms of Boolean algebra, and, conversely, it is very easy to draw a circuit whose action is specified by a given formula. Manipulation of expressions to show, for example, that two circuits are equivalent, is not so easy until some experience of working with Boolean algebra has been obtained. In the Harvard system about to be described all manipulation is carried out by ordinary algebraic processes so that the difficulty of working with unfamiliar rules is avoided. On the other hand, the working is often lengthy, and, since the symbolism does not represent quite so directly the action of a circuit, some manipulation of a formula may be necessary before a circuit having the action specified can be drawn.

In order to avoid confusion capital letters will be used to represent signals applied to gates, &c., when Harvard symbolism is being used. Since any symbol can take on the values 0 and 1 only we have

$$XX = X; \qquad (16)$$

this has the effect of simplifying expressions a good deal. If inputs X and Y are applied to a gate, the output of the gate is represented by XY. If X represents the input to an inverting circuit, the output is represented by $1 - X$, and it is convenient to use the abbreviation

$$X' = 1 - X.$$

If inputs X and Y are applied to an 'or' circuit the output is represented by $1 - X'Y'$, or $X + Y - XY$. The sum of X and Y, modulo 2, is given by $X + Y - 2XY$. It is easy to verify, by enumeration of all the possibilities, that these expressions take on only the values 0 and 1 and correctly represent the action of the elements concerned. Note that

$$\begin{aligned} XX' &= X(1 - X) \\ &= X - XX \\ &= 0. \end{aligned} \qquad \text{by (16)}$$

The adding circuit discussed above with the aid of Boolean notation is shown again in Fig. 6·7, this time analysed using the Harvard

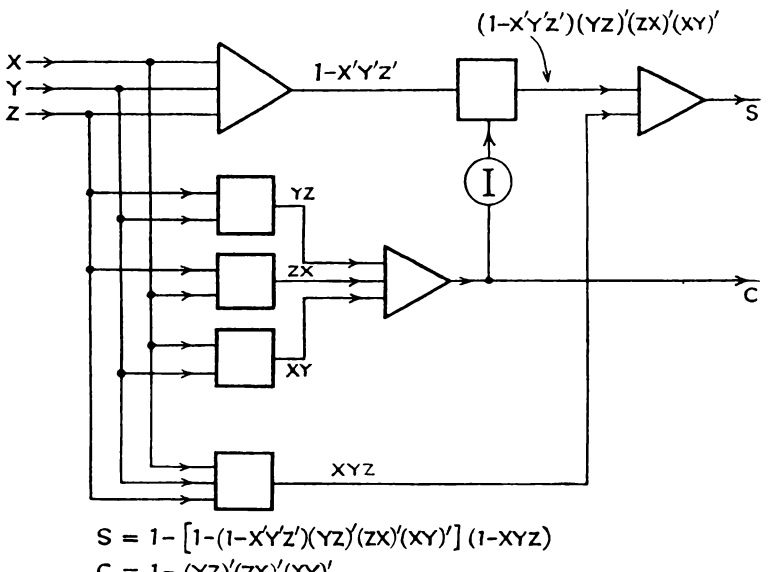


Fig. 6·7.—The adding circuit of Fig. 1·21 with annotations in Harvard algebraic notation.

symbolism. Reduction to their simplest terms of the expressions there given for the sum and carry outputs is carried out according to the rules of ordinary algebra. The reduction is straightforward although somewhat tedious. The results are as follows:

$$\begin{aligned} S &= 4XYZ - 2(YZ + ZX + XY) + X + Y + Z, \\ C &= YZ + ZX + XY - 2XYZ. \end{aligned}$$

The Practical Application of Switching Algebra

Circuit algebra can be a very useful aid in the design of efficient and economical switching circuits, but the stage has not yet been reached when it is possible to express in algebraic form the functions which a circuit is required to perform, and then to arrive at the most economical circuit by a process of algebraic manipulation alone. This is partly because the mathematical problem of reducing an algebraic expression, which represents the action of a circuit, to an equivalent form which, when directly interpreted in terms of relays or electronic components, gives the most economical design possible, has not been completely solved. One difficulty is that the definition of what is the most economical design depends on circumstances; for example, it might be desired to minimize the number of relay contacts or the number of relays, or, in an electronic circuit, the number of control grids or the number of diodes. Some progress has, however, been made with the formulation of rules for minimizing the number of terms in an expression representing a circuit with a single output; in particular, a system depending on the use of 'minimizing charts' has been successfully developed at Harvard for cases when there are no more than six input variables. A digital computer has also been programmed to carry out a similar task in the Boolean system (see reference 68).

A more fundamental limitation on the utility of switching algebra of the types discussed is that steady-state conditions only are con-

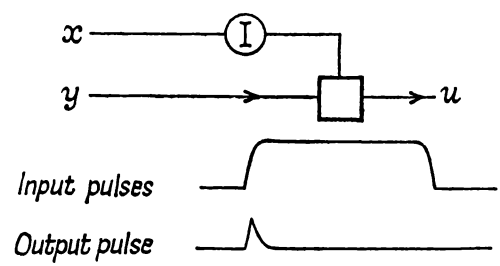


Fig. 6·8.—Imperfect switching action resulting from delay in an inverting circuit.

sidered; that is, no account is taken of what happens when a circuit is passing from one state to another. A common situation in which the blind application of switching algebra could lead to an erroneous conclusion because of the neglect of transient effects is illustrated in Fig. 6·8. Application of the ordinary methods of Boolean switching algebra to the circuit shown there gives

$$u = x'y,$$

so that if $x = 1$, $y = 1$, then $u = 0$. Thus if similar positive-going pulses, representing 1s, are applied simultaneously to the two inputs there

should be no output; in practice, however, the pulse passing through the inverting circuit will suffer a slight delay, and if this delay is comparable with the rise time of the pulses the gate may not close sufficiently rapidly to prevent the early part of the pulse applied to the y input from passing through. The result will be that a short 'spike' will appear on the output lead. This is illustrated in the lower part of the figure, where the form of the input pulses is also shown.

This defect in the operation of the circuit could have serious consequences. It may be removed in the manner shown in Fig. 6·9. Here

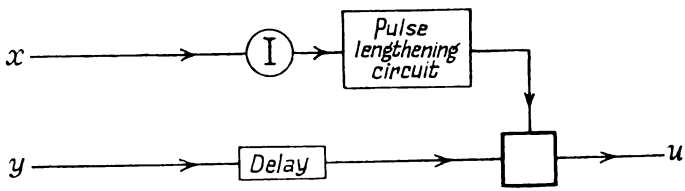


Fig. 6·9.—The circuit of Fig. 6·8 showing additional features required in a practical realization.

two additional features have been introduced; the pulse entering the gate has been delayed and the inverted pulse has been lengthened in order that the gate should be closed for the total duration of the applied pulse. It should be noted that these are both logical requirements and that the circuit could not be expected to work properly without them. They would not, however, normally be shown on a 'logical' diagram.

Similar considerations arise in connexion with relay circuits, since contacts which are nominally supposed to open and close together cannot be relied on to do so at exactly the same instant, even if they are mounted on the same relay. It is therefore necessary to scrutinize any circuit which has been designed with the help of Boolean algebra, and to modify it if transient conditions, which might give rise to faulty operation, are found to exist. An example of a case in which this was necessary will be found in reference 83. Switching algebras which take account of transient conditions have been formulated, but, up to the present time, have not been widely applied.

In addition to being slightly delayed, the pulses emerging from a switching circuit, such as an adder, are likely to be somewhat distorted and reduced in amplitude compared with the pulses entering the circuit. Since the satisfactory performance of many circuits depends on their being fed with pulses of standard shape and timing, it is desirable to provide means whereby pulses emerging from the various units of a machine can be standardized. This may be done by applying the pulses, amplified and slightly broadened, to one input of

a gate, the other input of which is fed with suitably phased clock pulses. This method of regeneration has already been described on p. 32 in connexion with ultrasonic storage. Since the pulse to be standardized has invariably suffered some delay, it cannot be used to control the passage of the clock pulse corresponding to its original timing. If the machine is provided with one set of clock pulses only, a pulse to be standardized must be retarded, by artificial delay introduced for the purpose, so as to occur slightly in advance of the succeeding clock pulse. It can then, after amplification and broadening, be used to control the passage of that pulse. The result is that the pulses emerging from the unit in question are delayed by exactly one pulse interval compared with the pulses which enter. Some machines are provided with two or three sets of differently phased clock pulses in order that the total delay need be no more than one-half or one-third of a pulse interval. Naturally, care must be taken when designing a machine to make sure that extra delay introduced for purposes of standardization is properly allowed for.

The broadening of pulses is not quite as easily effected as might be thought, since any attempt to use capacity or inductance for this purpose is likely to lead to a circuit whose performance depends on the duty cycle and on the shape of the original pulses. The necessity for broadening as a separate operation can be avoided if positive feedback is used in the manner shown in Fig. 6·10. Here the artificial delay introduced in front of the amplifier is so chosen that the pulses

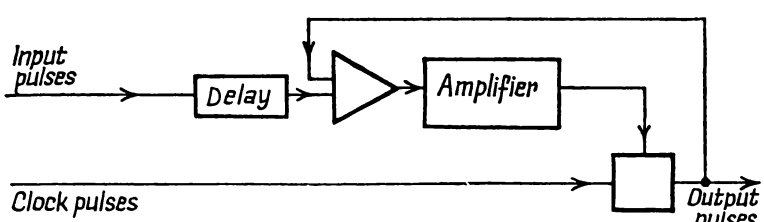


Fig. 6·10.—Clock pulse gating circuit with positive feedback.

arrive at the gate slightly in advance of the clock pulses; they need be long enough to cover the rising edges of the clock pulses only. Once the gate has been opened and a clock pulse has begun to pass through, a signal fed back from the output holds it open until the end of the clock pulse, even though the original pulse from the amplifier has died away. An equivalent system was used in the SEAC (reference 56).

Diode Switching Circuits

A few examples of the way in which thermionic valves may be used in the construction of switching circuits have already been given in

Chapter 1. Any further discussion of this subject would be outside the scope of this book, but something will be said about switching circuits based on the use of diodes.

Diodes have the property of passing current more readily in one direction than in the other, and an ideal diode would present zero

Fig. 6·11.—Schematic representation of a diode. The diode conducts if $V_2 > V_1$.

resistance to the flow of current one way and an infinite resistance to the flow of current the other way. Practical diodes fall short of this performance, and, as a result, switching circuits constructed with their aid inevitably allow some breakthrough of unwanted signals, the presence of which must be allowed for in the design of the subsequent circuits. While this is a matter of some importance to the engineer

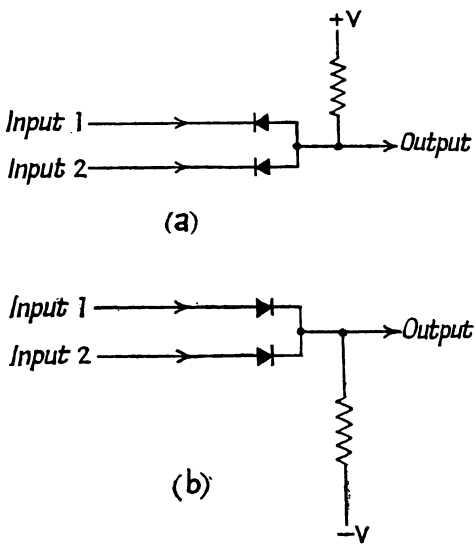


Fig. 6·12.—Simple diode switching circuits.

designing a practical machine, it is not usually considered under the heading of logical design. In what follows, therefore, it will be assumed that the diodes used approximate closely to the ideal. The notation used for an ideal diode is shown in Fig. 6·11. The diode conducts if $V_2 > V_1$ and does not conduct if $V_2 < V_1$.

A simple diode switching circuit is shown in Fig. 6·12 (*a*). If the two

inputs are both held at some fixed potential less than V the output will be held by the diodes at approximately the same potential. The output can rise appreciably in potential only if both inputs rise simultaneously. In the circuit shown in Fig. 6·12 (*b*), on the other hand, the output rises if either of the two inputs rises. If a high level of poten-

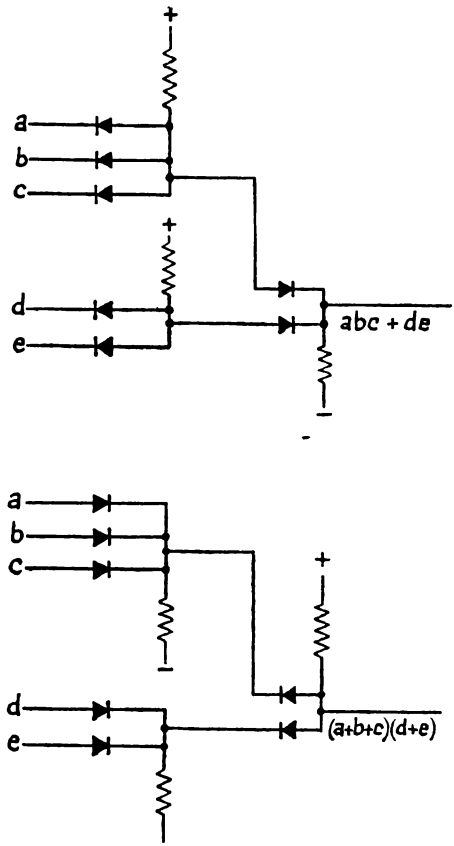


Fig. 6·13.—Examples of diode switching circuits; a high potential represents a 1.

tial represents a 1 and a low level a 0 the circuit of Fig. 6·12 (*a*) therefore functions as an 'and' circuit, and that of Fig. 6·12 (*b*) as an 'or' circuit. If, instead, a high level is taken to represent a 0 and a low level a 1 these roles are reversed, the circuit of Fig. 6·12 (*a*) functioning as an 'or' circuit, and that of Fig. 6·12 (*b*) as an 'and' circuit. This duality corresponds to the duality in Boolean algebra previously mentioned.

For best results a diode 'and' or a diode 'or' circuit should be fed from a low-impedance source, such as a cathode follower, and the output should be connected to a high-impedance destination, such as the

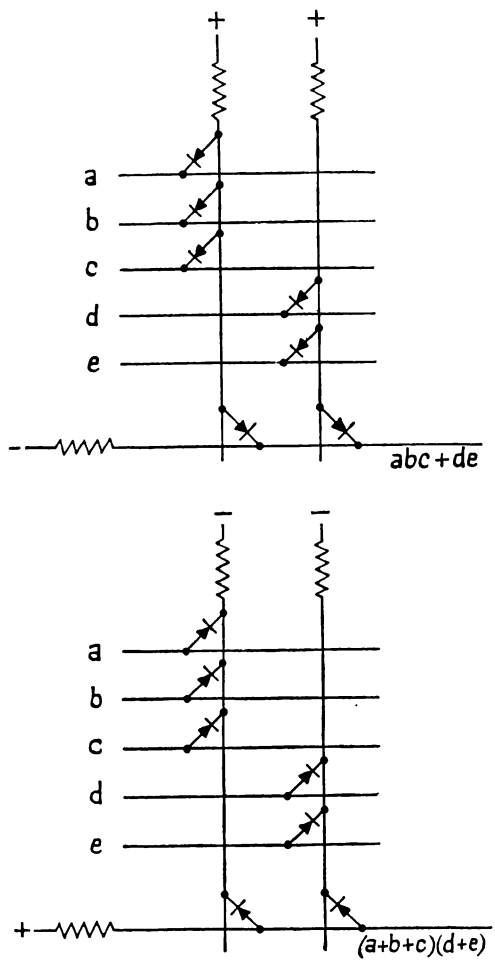


Fig. 6·14.—The circuits of Fig. 6·13 redrawn as diode matrices.

grid of a valve. However, if care is taken in the design it is possible to operate diode switching circuits in which a group of 'and' circuits are connected directly to an 'or' circuit, or in which a group of 'or' circuits are connected directly to an 'and' circuit. Examples are shown in Fig. 6·13. Such circuits are more usually drawn in the style

shown in Fig. 6·14, and are referred to as *diode matrices*. They are of some importance, and further examples will be encountered later in this chapter.

BINARY COMPUTING CIRCUITS

The method of constructing a binary adder shown in Fig. 6·6 is only one of a number which may be adopted, and several others will now be described. They will be derived from that of Fig. 6·6 by means of Boolean algebra, but the reader who prefers to do so may make a direct verification of the correctness of their action.

The expression for S given in equation (15) may be written

$$S=(x+y+z)C'+xyz.$$

Various transformations of this expression may be made. For example, we may write

$$S=(x+y+z)(C'+xyz),$$

corresponding to which is the adder shown in Fig. 6·15; again, by (11)

$$S=(x+y+z)[C(xyz)']',$$

which gives the adder shown in Fig. 6·16.

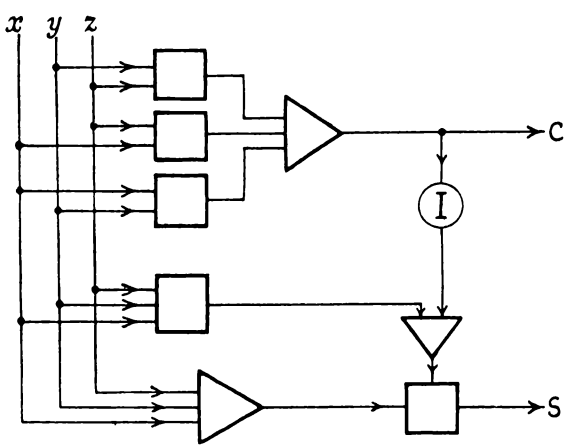


Fig. 6·15.—A binary adder.

The form of binary adder to be adopted in a given machine depends on the type of machine and on the electronic technique favoured by the designer. Naturally the main requirement is that the adder should contain as little equipment as is consistent with achieving the required speed of operation. If x', y', z' are available as input variables, in addition to x, y, z, a further series of adders are available

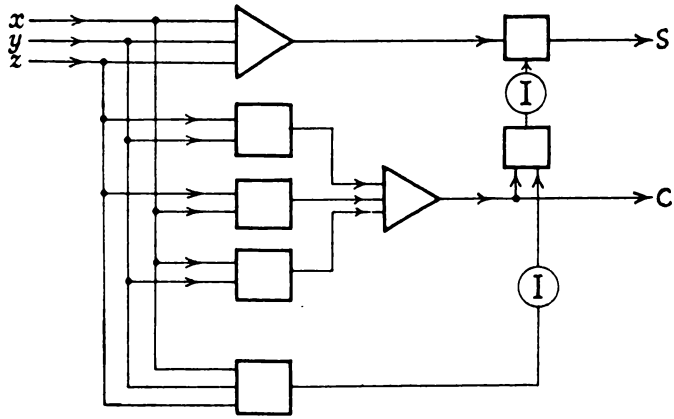


Fig. 6·16.—A binary adder.

for consideration; the most straightforward example is that shown in Fig. 6·17, which is based directly on the expressions

$$S = xy'z' + x'yz' + x'y'z + xyz,$$
$$C = yz + zx + xy.$$

This lends itself to construction as a diode matrix as shown in Fig. 6·18.

It may be noted that a three-input binary adder has the property of being self-dual; i.e.,

$$S'(x, y, z) = S(x', y', z'),$$
$$C'(x, y, z) = C(x', y', z').$$

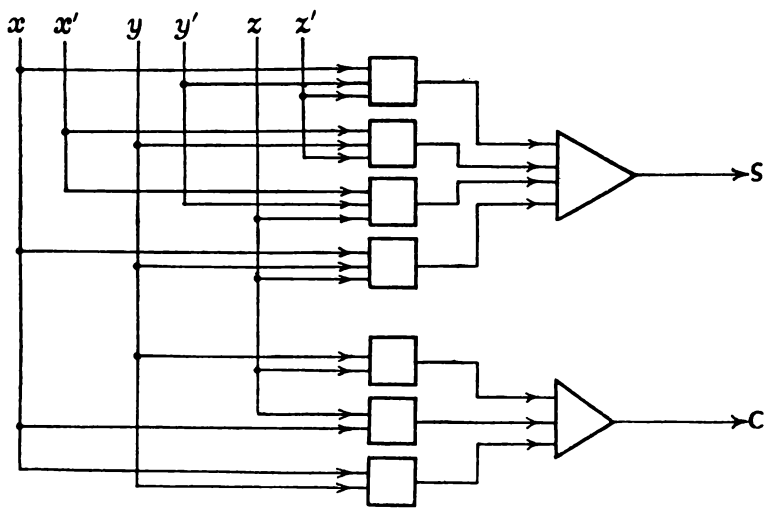


Fig. 6·17.—A binary adder.

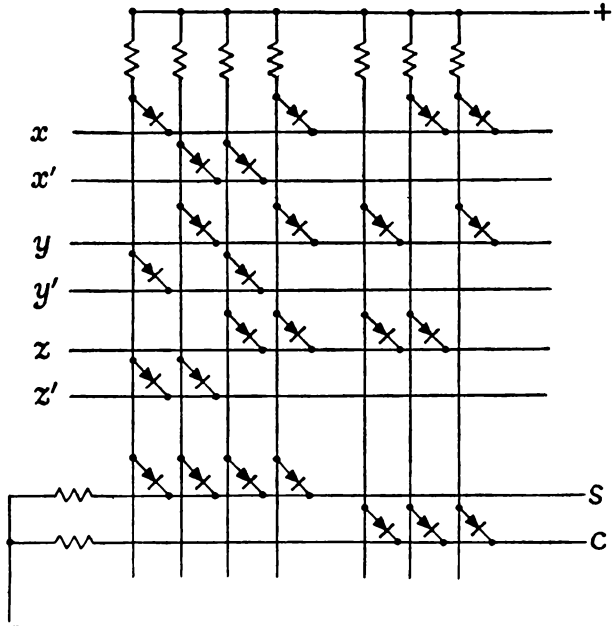


Fig. 6·18.—A binary adder in the form of a diode matrix. 1s are represented by high potentials and 0s by low potentials.

The reader may also verify the following identities:

$$S(x, y', z') = S(x', y, z') = S(x', y', z) = S(x, y, z).$$

Two-input Adders

Fig. 6·19 shows a two-input adder circuit capable of accepting two binary inputs and giving out the appropriate sum and carry digits, namely

$$s = xy' + x'y, \quad c = xy.$$

An alternative form is shown in Fig. 6·20.

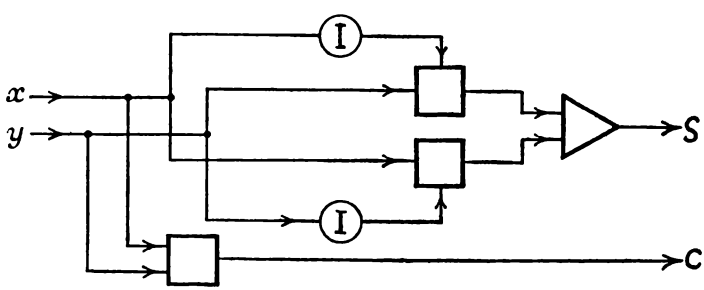


Fig. 6·19.—A two-input adder, or half-adder.

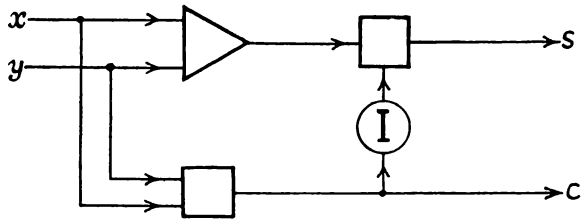


Fig. 6·20.—An alternative half-adder.

Two-input adders are of interest because a pair may be combined in the manner shown in Fig. 6·21 to form a three-input adder; they are, for this reason, frequently referred to as *half-adders*.

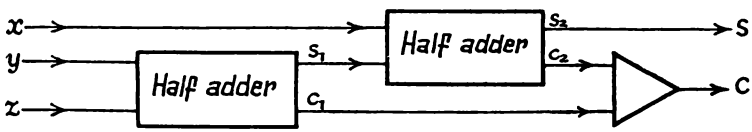


Fig. 6·21.—Two half-adders connected to form a complete adder.

That the arrangement of Fig. 6·21 constitutes a three-input adder should become obvious when it is observed that, since either s_1 or c_1 must be 0, the carry digits from the two adders cannot both be 1s. Alternatively, the same fact may be established by Boolean algebra as follows:

$$
\begin{aligned}
s_2 &= s_1x' + s_1'x \\
&= (yz' + y'z)x' + (yz' + y'z)'x \\
&= (yz' + y'z)x' + (yz')'(y'z)'x && \text{by (11)} \\
&= (yz' + y'z)x' + (y' + z)(y + z')x && \text{by (12)} \\
&= xy'z' + x'yz' + x'y'z + xyz && \text{by (14)} \\
&= S(x, y, z).
\end{aligned}
$$

$$
\begin{aligned}
c_1 + c_2 &= yz + xs_1 \\
&= yz + x(yz' + y'z) \\
&= yz + yz + x(yz' + y'z) && \text{by (6)} \\
&= y(z + xz') + z(y + xy') \\
&= y(z + x) + z(y + x) && \text{by (9)} \\
&= yz + zx + xy && \text{by (6)} \\
&= C(x, y, z).
\end{aligned}
$$

Binary Subtractors

Subtractors are not of the same importance as adders, since it is usually more convenient to make use of complements in conjunction

with an adder than to perform subtraction directly. The following table gives the difference output, D, and the carry output, K, for a three-input binary subtractor. x is the resident digit, y the incident digit, and z the incoming carry digit.

x	y	z	D	K
0	0	0	0	0
1	0	0	1	0
0	1	0	1	1
0	0	1	1	1
1	1	0	0	0
1	0	1	0	0
0	1	1	0	1
1	1	1	1	1

It will be observed that

$$D = S(x, y, z); \quad K = C(x', y, z).$$

The design of subtractors is thus closely related to that of adders, and combined adder–subtractors can be designed. The connexion between subtraction and addition of complements is shown by writing the equations in the form (see p. 224)

$$D = S(x, y', z'); \quad K' = C(x, y', z').$$

Translating Circuits

The binary adders which have been described are examples of *translating* or *coding* circuits. As will be seen later, decimal adders can be designed in a similar way. Translating circuits have various other applications in digital computers, particularly in control systems. The design of translating circuits may be facilitated by the use of switching algebra.

A translating circuit accepts a number (say n) of binary inputs and provides a number (not necessarily equal to n) of binary outputs. Each output is a uniquely determined function of the n input variables. The function corresponding to a particular output is called the *switching function* for that output. In the case of a translating circuit with three binary inputs and one binary output, the switching function may be defined by a table of the following form:

x	y	z	f
0	0	0	f_0
0	0	1	f_1
0	1	0	f_2
0	1	1	f_3
1	0	0	f_4
	.		.
	.		.
	.		.
1	1	1	f_7

If a numerical value (0 or 1) is given to each of the quantities f_0, f_1, &c., the action of the circuit is uniquely specified. The switching function f may be written:

$$f = x'y'z'f_0 + x'y'zf_1 + x'yz'f_2 + \ldots + xyzf_7;$$

for example, the switching function corresponding to the sum output of a binary adder (cf. Table 1·2 on p. 34) is defined by

$$f_1 = f_2 = f_4 = f_7 = 1; \quad f_0 = f_3 = f_5 = f_6 = 0.$$

Switching functions written down in the above manner can often be simplified by Boolean manipulation. As an example, suppose it is desired to design a circuit which will accept the separate digits x, y, z, t of a four-digit binary number and give a zero output if the number represents a decimal digit, i.e., if it is less than ten, and a unit output if it does not represent a decimal digit, i.e., if it is greater than or equal to ten. The switching function for the output of the circuit may be written down and simplified as follows:

$$\begin{aligned} f &= xy'zt' + xy'zt + xyz't' + xyz't + xyzt' + xyzt \\ &= xy'z + xyz' + xyz \\ &= xy'z + xy \\ &= x(y'z + y) \\ &= x(y + z). \end{aligned}$$

The logical diagram of a circuit meeting this specification is shown in Fig. 6·22.

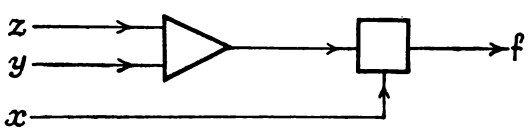


Fig. 6·22.—Circuit for generating $f = x(y+z)$.

As a further example, consider the design of circuits for translating decimal digits, expressed in binary coded decimal form, to a two-out-of-five code. The following table shows the translation to be effected:

Input				Output				
x	y	z	t	a	b	c	d	e
0	0	0	0	0	0	0	1	1
0	0	0	1	1	1	0	0	0
0	0	1	0	0	0	1	1	0
0	0	1	1	1	0	0	1	0
0	1	0	0	1	0	0	0	1
0	1	0	1	0	1	0	1	0
0	1	1	0	1	0	1	0	0
0	1	1	1	0	0	1	0	1
1	0	0	0	0	1	0	0	1
1	0	0	1	0	1	1	0	0

Expressions for a, b, c, d, e in terms of x, y, z, t are

$$\begin{aligned}
a &= x'y'z't + x'y'zt + x'yz't' + x'yzt' \\
&= x'y't + x'yt' \\
&= x'(yt' + y't); \\
b &= x'y'z't + x'yz't + xy'z't' + xy'z't \\
&= x'z't + xy'z' \\
&= z'(x't + xy'); \\
c &= x'y'zt' + x'yzt' + x'yzt + xy'z't \\
&= x'zt' + x'yz + xy'z't \\
&= x'z(t' + y) + xy'z't; \\
d &= x'y'z't' + x'y'zt' + x'y'zt + x'yz't \\
&= x'y'z + x'yz't + x'y'z't' \\
&= x'(y'z + yz't + y'z't'); \\
e &= x'y'z't' + x'yz't' + x'yzt + xy'z't' \\
&= y'z't' + x'yz't' + x'yzt \\
&= x'z't' + y'z't' + x'yzt \\
&= z't'(x' + y') + x'yzt.
\end{aligned}$$

A translating unit designed in accordance with the above equations will give the values of a, b, c, d, e shown in the table when the input digits $xyzt$ represent a decimal digit; it will give $a = b = c = d = e = 0$ otherwise. This feature may be of value in connexion with error detection. If it is not insisted on, the expressions for a, b, c, d, e obtained above may be simplified further by making use of the condition that $xyzt$ should represent a decimal digit, namely:

$$x(y + z) = 0,$$

i.e.,

$$xy = xz = 0,$$

from which it follows, by taking the dual, that

$$\begin{aligned}
x' + y' &= 1, \\
x' + z' &= 1.
\end{aligned}$$

By multiplying these equations by x and y, and by x and z, respectively, it further follows that

$$\begin{aligned}
xy' &= x, & x'y &= y, \\
xz' &= x, & x'z &= z.
\end{aligned}$$

We then have

$$\begin{aligned}
a &= yt' + x'y't; \\
b &= z'(x't + x) \\
&= z'(t + x) \\
&= x + z't; \\
c &= zt' + yz + xt; \\
d &= y'z + yz't + x'y'z't'; \\
e &= z't' + yzt.
\end{aligned}$$

Binary Accumulators and Counters

If the carry output of a three-input adder is connected to one of the inputs through a delay circuit a serial adder is obtained. This may be combined with a serial storage device, in the manner described in Chapter 2, to form a serial accumulator. In a similar way, a parallel accumulator may be formed from a static register and a group of adders, one for each binary digit. If the addition time in a parallel machine is to be short, it is important to choose a type of adder in which the carry pulses are passed through as few gates and other circuits as possible when being propagated along the length of the register, and to give special attention to the design of these gates and circuits so as to make them fast in operation.

An accumulator can be used as a counter if 1 is added to the number stored in it each time a count is recorded. It is sufficient, however,

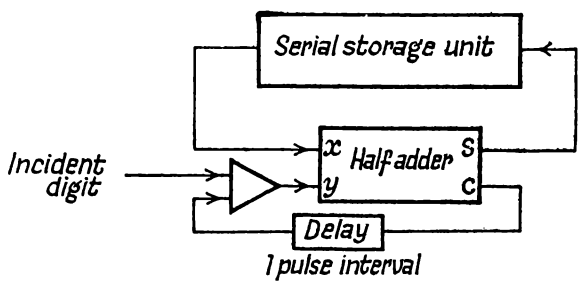


Fig. 6·23.—A serial binary counter.

if the accumulator has a half-adder in each stage instead of a complete adder. Fig. 6·23 shows a serial counter designed in this way. It will easily be seen that the single digit of the incident number must necessarily arrive at the input to the adder one pulse interval in advance of the first of any carry digits generated by the adder.

A better-known type of counter is that based on the use of a bistable circuit, which can be caused to pass from the state it happens to be in to the other state by the application of a pulse of suitable amplitude and duration. An example of such a circuit is shown in Fig. 6·24; this resembles an ordinary flip-flop, except that the input pulses are applied, through buffer diodes, to both grids, and a resistor shunted by a condenser is included in each cathode circuit.

When the circuit is in a quiescent condition one valve, say V_a, is conducting and the other, V_b, non-conducting. The cathode of V_a is, therefore, at a positive potential with respect to earth while the cathode of V_b is at earth potential. The circuit is caused to change its state by the application to the input terminal of a negative-going

pulse of sufficient amplitude to render both valves non-conducting. The duration of this pulse is short compared with the time constants of the circuits formed by the resistors and condensers in the cathode circuits of the valves; the cathode of V_a therefore remains at a positive potential with respect to earth. When the applied pulse has died away both grids begin to rise in potential as the stray capacities associated with them are charged through the resistors R_a and R_b. When the potential of the grid of V_b approaches earth potential this valve

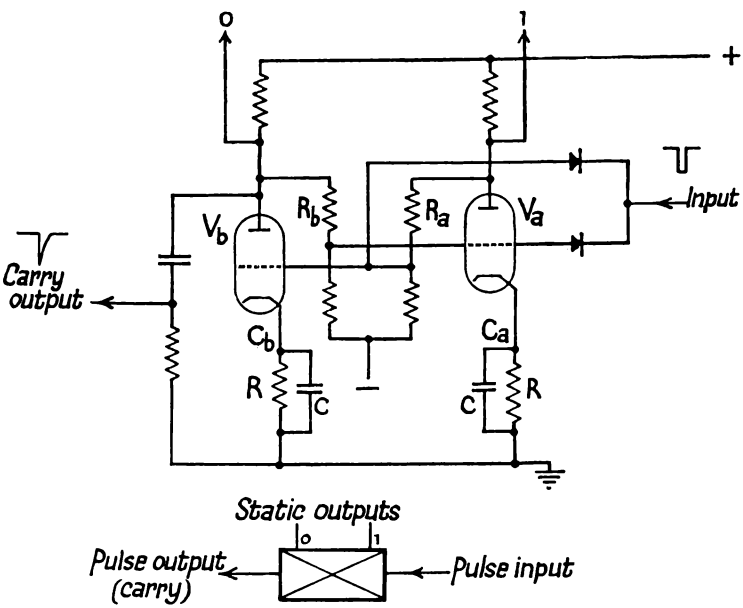


Fig. 6·24.—A bi-stable counter circuit; 1s are represented by negative-going pulses.

begins to conduct; V_a, on the other hand, has no tendency to conduct, since its cathode is still at a positive potential with respect to earth. As soon as V_b begins to conduct its anode potential falls, with the result that the rise in potential of the grid of V_a is slowed down and ultimately arrested. The circuit ultimately becomes quiescent with V_b in a conducting condition and V_a non-conducting. The anode of V_b is connected through a circuit with a short time constant to the carry output lead. A short negative-going pulse appears on this lead when V_b changes from a non-conducting to a conducting condition.

A multi-stage counter may be formed by connecting in cascade a number of circuits of the type just shown, the carry output of one

stage being connected to the input terminal of the succeeding stage. Each stage is said to be storing a 1 when the valve corresponding to V_a is conducting and a 0 when that corresponding to V_b is conducting, and it will be seen that whenever the digit stored in a given stage changes from a 1 to a 0 a negative-going pulse is passed to the succeeding stage. A positive-going pulse appears on the carry output lead when the digit changes from a 0 to a 1, but this is without effect

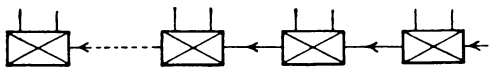


Fig. 6·25.—A multi-stage counter using bi-stable counting circuits.

on the succeeding stage. A schematic notation for a single-stage counter is shown in the lower part of Fig. 6·24. Fig. 6·25 shows a multi-stage counter drawn with this notation.

In the bi-stable counting circuit just described the cathode condensers provide temporary storage for the resident digits during the time the pulses representing the incident digits are being applied. This makes it possible to construct a counter with one bi-stable circuit per stage instead of the two required in a purely 'logical' counter. It is necessary, however, that the size and shape of the applied pulses should fall within certain limits, and some designers are of the opinion that such circuits are likely to prove more troublesome in operation than purely logical ones.

The circuit of Fig. 6·25 may be converted into an accumulator by providing an input to each stage in the manner shown in Fig. 6·26.

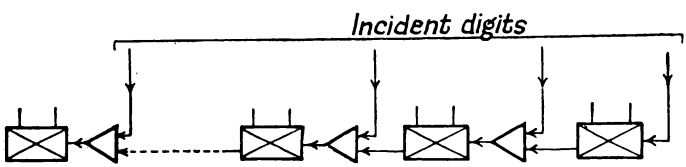


Fig. 6·26.—An accumulator with successive carry using bi stable counting circuits.

The digits of the incident number are applied one at a time, to the leads shown, in the form of pulses of suitable shape and size. A sufficient interval must elapse after the application of each pulse to enable a carry to be propagated, if necessary, to the end of the register. This is probably the simplest accumulator which can be devised, but it is not particularly fast in view of the fact that successive carry (see p. 14) is used. The great simplicity of this form of accumulato rarises from the fact that a single bi-stable counter records automatically the

sum (modulo 2) of the pulses applied to it, and can also be made to give a pulse output when this sum changes from 1 to 0.

Bi-stable counters can also be used in accumulator circuits in which greater speed is obtained by the use of anticipatory carry. However, since in these circuits the computation of the sum of three digits, modulo 2, is required, the degree of simplification obtainable is not so great. It is necessary to provide some form of two-input adding circuit to form the sum modulo 2 of the incident and incoming carry digits; the summation is completed by applying this sum to the bi-stable counter which holds the resident digit. There must be additional circuit elements for forming the outgoing carry digit. A logical

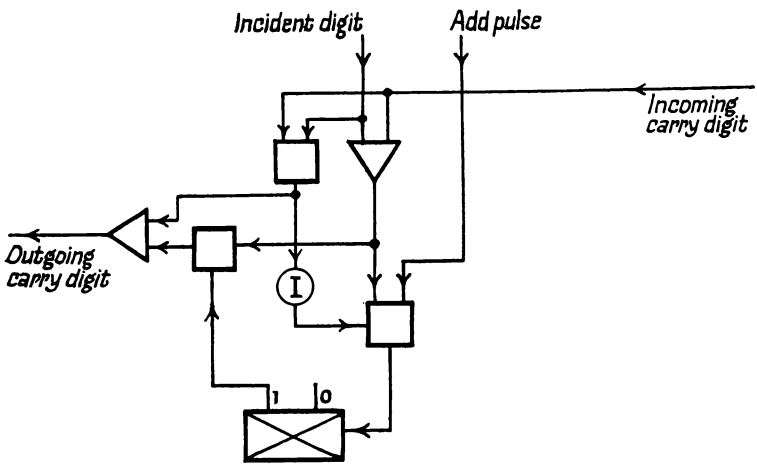


Fig. 6·27.—Single stage of an accumulator with anticipatory carry using bi-stable counting circuits.

diagram of one stage of an accumulator based on these principles is shown in Fig. 6·27; this arrangement is very similar to that used in the SWAC. Wherever an addition is to be performed the digits of the incident number are presented to the accumulator in static form; sufficient time is then allowed to elapse for the carry circuits to settle down (allowing for a carry to be propagated, if necessary, from one end of the register to the other), and the addition is completed by applying a pulse to the line marked 'add pulse'. This pulse reaches all stages simultaneously.

Kirchhoff Switching Circuits

A type of switching circuit not so far discussed is based on the use of a resistance network followed by a level-discriminating circuit.

Fig. 6·28 shows a two-input gate constructed on this principle. The input terminals are at potential 0 or V according as the corresponding input variable is 0 or 1. The input to the level-discriminating circuit therefore takes up one of the following values: 0, $rV/(R+r)$

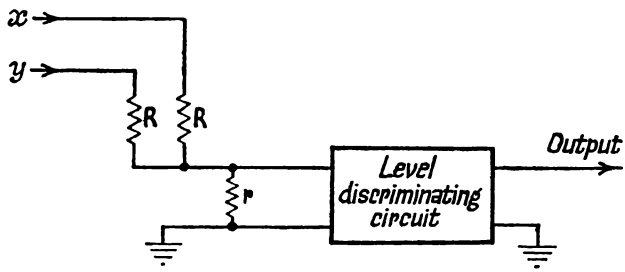


Fig. 6·28.—A Kirchhoff two-input gate.

$rV/(\frac{1}{2}R+r)$, or approximately, if $R \gg r$, 0, rV/R, $2rV/R$. The level-discriminating circuit gives zero output unless the input exceeds r/R by a suitable margin, in which case it gives an output V'. It is easily seen that, if the output V' is taken to represent 1 and zero output to represent 0, the circuit operates as a gate. Similar circuits with more inputs can be constructed; a three-input circuit, for example, is shown

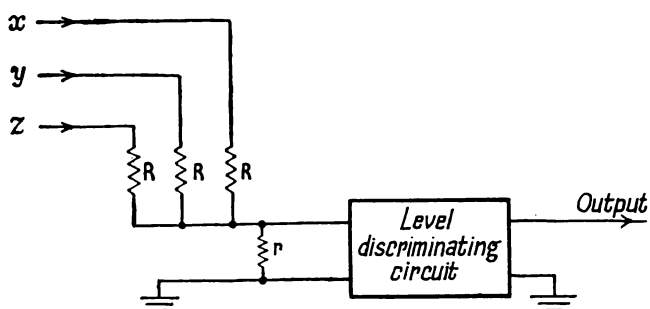


Fig. 6·29.—A Kirchhoff three-input gate.

in Fig. 6·29. Here the discriminator can be set so as to give an output either when at least one of the inputs is 1 or, alternatively, when two or more inputs are 1. In the latter case the circuit does not correspond to any of the basic logical elements which have so far been discussed, and to obtain the same effect with these elements a circuit containing three gates and an 'or' circuit would be required.

An adder based on the use of two resistance networks, each followed by a discriminating circuit, is shown in Fig. 6·30. One of the resistance networks has three inputs, and the associated discriminat-

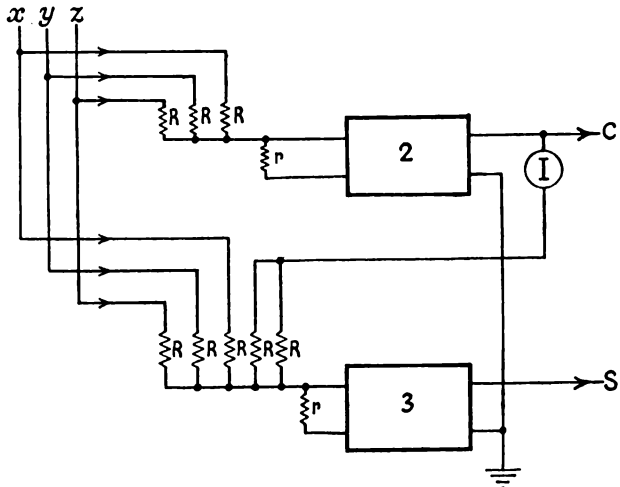


Fig. 6·30.—A schematic diagram of a Kirchhoff adder.

ing circuit gives a non-zero output when at least two of the inputs are 1; the other resistance network has five inputs, and the discriminating circuit gives a non-zero output when at least three of these inputs are 1s. This diagram is schematic only, and a complete circuit diagram of the adder is shown in Fig. 6·31. Adders working on this

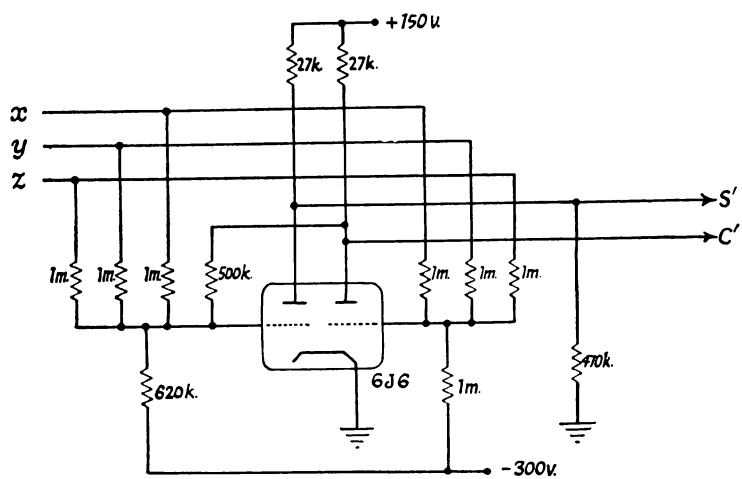


Fig. 6·31.—Circuit diagram of a Kirchhoff adder. The voltages on the input lines swing between 40 volts (representing a 0) and 140 volts (representing a 1). The output voltages S′ and C′ swing between the the same limits.

principle are known as Kirchhoff adders, and have the advantage of simplicity. They require, however, the use of close-tolerance resistors, and the signals applied to them must be standardized within close limits. It may be necessary to effect this standardization by connecting limiting diodes to the inputs, and this detracts somewhat from the simplicity of the circuits. Moreover, for a given operating margin and power consumption, Kirchhoff adders tend to be rather slower in operation than logical adders on account of the fairly long time constant which is usually associated with resistance networks. In view of these disadvantages, and of the fact that the saving of a few valves in the adding circuits does not greatly reduce the amount of equipment in a large computer, Kirchhoff adding circuits are not now much in favour among designers of large machines.

A Kirchhoff decoding tree capable of accepting two binary inputs is shown in Fig. 6·32. In this case the level-discriminating circuit must have four outputs, one for each combination of the input variables. A convenient circuit consists of four valves with a common cathode resistor, the outputs being taken from the anodes as shown in the diagram. Decoding trees of this type, with as many as a hundred outputs, were used in the function tables of the ENIAC, and similar circuits were later used for order decoding and storage access purposes

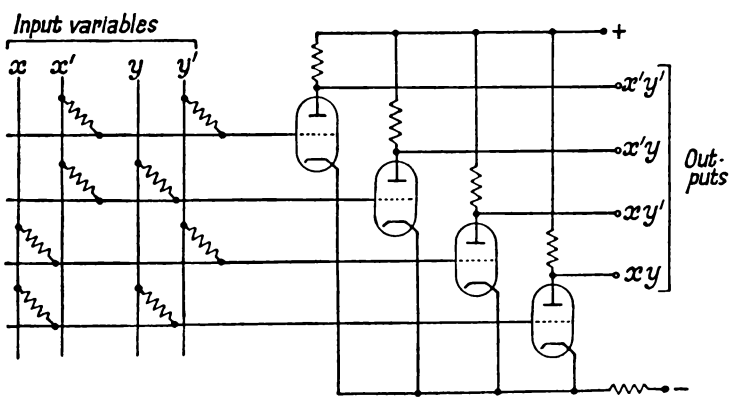


Fig. 6·32.—A Kirchhoff decoding tree.

in various binary computers. They suffer, however, from the same disadvantages as Kirchhoff adders, and now that cheap and stable germanium diodes have become available most designers prefer to use diode matrices.

Binary Multipliers

Some remarks have already been made in Chapter 2 on the logical design of binary multiplying circuits for both serial and parallel ma-

chines. In a serial machine greater multiplication speed may be obtained by using more than the single adder that is essential. Fig. 6·33 shows the logical diagram of a fast serial multiplier which uses one less adder than there are binary digits in the numbers. The digits of the multiplier are presented to the adder in static form from a register consisting of a set of flip-flops; the multiplicand arrives in dynamic form as a train of pulses, and the product is emitted in the same form.

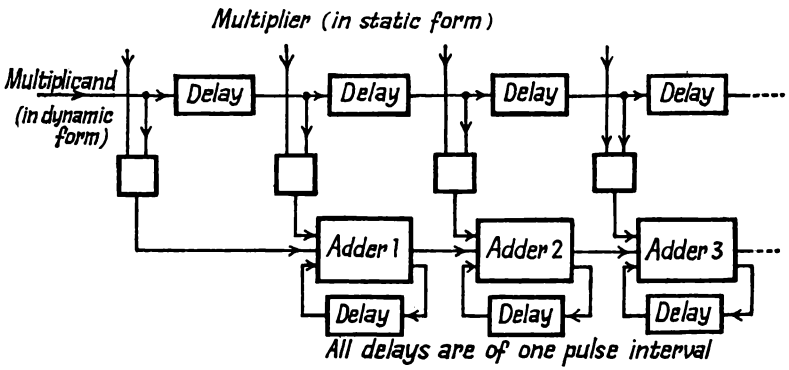


Fig. 6·33.—A fast serial multiplier.

The circuit shown is capable of generating the product in approximately two number times.

Variants of the circuit shown in Fig. 6·33 may also be devised. For example, in a machine built by Ferranti Ltd. the multiplication is done in two stages; this halves the number of adders required, but approximately doubles the multiplication time. If multi-input adders are available (that is, adders capable of accepting four or more binary digits), the number of adders may be reduced appreciably. Unfortunately multi-input adders are not simply constructed on logical principles, and the scheme would appear to be attractive only in systems in which Kirchhoff adders can be regarded as acceptable.

Binary Dividers

The division of a positive number, y, held in an accumulator by another positive number, x (greater than y), held in a register may be accomplished by proceeding according to the following rules. (It is assumed that a further register is available in which the binary digits of the quotient may be recorded one by one.) Subtract x from the accumulator. If the result is positive or zero, record a 1 in the quotient register; if the result is negative, record a 0 and add x into the accumulator so as to restore the original content. Shift the content of the accumulator one place to the left and repeat the procedure. In

this way the digits of the quotient are built up one by one in the quotient register, starting with the most significant. The binary point is to the right of the first recorded digit.

In a parallel machine it is possible to test the sign of the output from the adder before deciding whether or not that output is to be accepted. The subtraction of x from the accumulator may, therefore, be prevented from taking place if the result would be negative, so that the subsequent restoration by adding x may be avoided. This enables the division to be speeded up.

In the method described in the first paragraph of this section, whenever the number in the accumulator is found to be negative, x is added into the accumulator, the content is shifted one place to the left, and x is subtracted. This is equivalent to shifting the content of the accumulator one place to the left and adding x. An alternative procedure is, therefore, as follows. Examine the number in the accumulator. If this is positive or zero, subtract x; if it is negative, add x. Shift the number in the accumulator one place to the left and repeat. Whenever, after an addition or a subtraction, the number in the accumulator is found to be positive or zero, record a 1 in the quotient register; otherwise record a 0. This procedure may be used in either serial or parallel machines.

The procedure just described can easily be modified to deal with the situation in which either the divisor, or the dividend, or both, may be negative numbers. It then takes the following form. Examine the number in the accumulator. If this is zero or has the same sign as x, subtract x; otherwise add x. Shift one place to the left and repeat. Whenever, after an addition or subtraction, the number in the accumulator is found to be zero or to have the same sign as x, record a 1; otherwise record a 0. The binary point follows immediately after the first recorded digit. When the quotient is negative it appears in the quotient register in complementary form.

It will be observed that the quotient register need have no facilities for addition associated with it. However, the ordinary procedure for rounding off, namely the addition of a 1 in a digital position one place to the right of the last digit to be retained, is not then applicable. An alternative method, which has been adopted for the machine constructed at the Institute for Advanced Study, is to force the last digit retained in the quotient to be a 1 in all circumstances. This procedure introduces no bias, since it has the effect of increasing the truncated quotient by one unit in the last place on half the occasions on which it is applied; the root-mean-square error introduced is, however, twice as much as that introduced by the conventional rounding-off process, amounting to $1/\sqrt{3}$ of a unit in the last place retained.

The operation of taking a square root resembles that of division,

and it is not difficult to adapt a divider, by adding a certain amount of additional equipment, so as to make it capable of extracting square roots.

DECIMAL COMPUTING CIRCUITS

An introductory discussion of the problems confronting the designer of an electronic decimal computing machine was given in Chapter 2. A distinction was there drawn between true decimal machines, which make use of elements with ten mutually exclusive stable conditions, and machines which are constructed from binary elements and make use of some form of binary coding for decimal digits.

The demand for pulse counters for use in nuclear and atomic physics has led to the development of a number of decimal counting tubes. These are mostly gas-discharge tubes in which the discharge can take place in one of ten compartments and can be stepped round by applying pulses to appropriate electrodes. These tubes have not, however, found the wide application in decimal computing circuits which at first sight might have been expected for them. This is partly because they can be operated at relatively slow speeds only, on account of the de-ionization time of the gas. It is doubtful, however, whether the situation would be much changed if fast vacuum tubes, having similar properties, became generally available—various such tubes have been produced experimentally—because of the problem of providing a large high-speed store of a true decimal character. A small machine using Dekatrons—a form of ten-position gas tube—for both arithmetic unit and store has been constructed at A.E.R.E., Harwell. The total high-speed storage capacity is, however, only 320 decimal digits, and a really large store composed of Dekatrons—with one Dekatron for each decimal digit stored—would not be an economic proposition. All the successful high-speed stores so far proposed are binary in operation and can only be used to store decimal digits when these are expressed in a binary code. If this is done it is more natural and convenient to use binary-coded circuits in the arithmetic unit as well.

For the reasons just given, large-scale electronic computing machines working in the decimal system depend on the use of a binary code to express the decimal digits. The choice of this code is all-important. The object is to choose a code which will enable the machine to be designed with the greatest possible economy of material. It may be desirable to use different codes in different parts of the machine, since some codes are particularly efficient for storage, some provide useful checking facilities, and others lend themselves to the economical design of adders or multipliers. The choice of a code and the design of coded decimal computing circuits involve a whole field

of technique which is not encountered by the designer of a binary machine. The subject has a considerable theoretical interest in its own right, and much work remains to be done before a wholly systematic presentation can be attempted. The present account is based largely on the publications of the Computation Laboratory of Harvard University.

The codes which lead to the most economical use of storage capacity are those in which four binary digits are used to represent each decimal digit. There are 16 distinct code groups with four binary digits, and the number of ways of choosing 10 of these to represent the decimal digit is 16!/6!, which is approximately equal to $2{\cdot}9.10^{10}$.

A given coding scheme may be transformed into another (usually distinct) coding scheme, by interchanging one or more pairs of digits, or by replacing certain digits by their inverses, or by a combination of these things. Such a transformation is known as an *input transformation*. It may be applied to information carried in parallel form on a set of wires simply by crossing the wires over to interchange the variables and introducing inverting valves where necessary. Similarly, information expressed in a given code and stored in a register consisting of flip-flops may be read out in another code, related to the first by an input transformation, by taking output connexions from the appropriate sides of the several flip-flops and having regard to the order in which the outputs are arranged. Thus in a parallel machine the application of an input transformation is a trivial matter. In a serial machine it is usually necessary that the information should first be converted to parallel form before an input transformation can be applied. Input transformations are the only transformations which can be made without the use of more or less complicated translating circuits.

Since four variables may be arranged in 4! permutations, and in each permutation any variable may be replaced by its complement, there are in all $2^4.4!$, that is 384, input transformations of four variables. It can easily be verified that no system of coding the 10 decimal digits, using four binary digits, is invariant under any input transformation. It follows that there are 16!/384.6! or (1,001/48).10! essentially distinct decimal coding systems from which others may be derived by means of input transformations. This number is approximately $7{\cdot}6.10^7$.

Table 6·1 gives some of the four-digit codes which have been used, or considered for use, in automatic computers. Columns (*a*) and (*b*) give the straight binary code and the Stibitz excess-three code, which have already been discussed in Chapter 2. The straight binary code is an example of a code in which the binary digits carry constant values, or weights, whenever they occur; in this case the weights are 8, 4, 2, 1.

There are only 17 distinct codes with constant positive weights, but they are of primary importance in connexion with the design of computers, and several of them appear in the table.

TABLE 6·1

Binary code groups	Decimal digits assigned							
	(*a*) 8, 4, 2, 1	(*b*)	(*c*) 8, 4, −2, −1	(*d*) 2*, 4, 2, 1	(*e*) 7, 4, 2, 1	(*f*)	(*g*)	(*h*)
0 0 0 0	0	—	0	0	0	—	—	—
0 0 0 1	1	—	—	1	1	0	1	1
0 0 1 0	2	—	—	2	2	1	2	5
0 0 1 1	3	0	—	3	3	2	6	0
0 1 0 0	4	1	4	4	4	3	4	7
0 1 0 1	5	2	3	—	5	4	0	9
0 1 1 0	6	3	2	—	6	5	5	4
0 1 1 1	7	4	1	—	—	—	—	—
1 0 0 0	8	5	8	—	7	6	8	2
1 0 0 1	9	6	7	—	8	7	3	6
1 0 1 0	—	7	6	—	9	8	7	8
1 0 1 1	—	8	5	5	—	—	—	—
1 1 0 0	—	9	—	6	—	9	9	3
1 1 0 1	—	—	—	7	—	—	—	—
1 1 1 0	—	—	—	8	—	—	—	—
1 1 1 1	—	—	9	9	—	—	—	—

The excess-three code is an example of a code in which complements on 9 can be obtained by simple inversion of the binary digits. There are altogether ${}^{8}C_{5}.2^{5}.5! \simeq 2{\cdot}2.10^{5}$ codes with this property. Another example is the constant-weight code proposed by Rubinoff given in column (*c*); this can be derived from the excess-three code by the input transformation $wxyz \rightarrow wxy'z'$; note that in this code two of the weights are negative. Another constant-weight code in which complements may be obtained by inversion is given in column (*d*); here two of the weights have the same value, namely 2, and a star is used to differentiate them. The code was used in the function table and card-reading circuits of the ENIAC and in the Harvard Mark III machine.

The code given in column (*e*) makes use of code groups each containing two 1s or less, and minimizes the average number of 1s required to represent a decimal number, and hence the power required to operate a mechanical device. For a similar reason it has found applications in index systems in which cards are selected by passing knitting-needles through holes punched in them. The code given in column (*f*) makes use of the ten code groups which contain one or two 1s, and is useful in certain electronic storage and computing circuits in which it is advantageous to restrict the fluctuation of the number of 1s as the numbers change. The rearrangement of this

code, given in column (*g*), has important additional properties, connected with input transformations, which make it of value in the design of decimal multiplying circuits. A further rearrangement, shown in column (*h*), is useful in the design of decimal adders (see pp. 246–9).

If an extra digit is added to each of the ten code groups shown in column (*f*), (*g*) or (*h*) a two-out-of-five code may be constructed. The extra digit must be chosen to be a 0 or a 1 in such a way that each group contains two, and only two, 1s. The code resulting from column (*h*) is as follows:

0	00011	5	10010
1	10001	6	01001
2	11000	7	10100
3	01100	8	01010
4	00110	9	00101

The value of such a code in giving security against the introduction of errors has already been mentioned in Chapter 2 in connexion with the output system of the EDSAC. This security arises from the fact that two digits must be changed in order to convert a code group representing a particular decimal digit into one representing another decimal digit, and, moreover, the change must be in one case from a 0 to a 1 and in the other from a 1 to a 0. There are exactly ten five-digit code groups each having two 1s and three 0s, and from these 10!/5! or 30,240 essentially distinct coding systems can be formed. All other two-out-of-five codes may be obtained from these by the application of input transformations.

Decimal Adding Circuits and Accumulators

A clear distinction may be drawn between those decimal computing circuits in which addition is performed by counting and those in which it is performed by coding. In the former the functions of counting and storage are closely bound up together, and an accumulator must be regarded as a single unit rather than as the combination of an adding and storage device. An accumulator of this type is distinguished by the fact that during addition it passes through a sequence of distinct states intermediate between its original and final states. Desk calculating machines, without exception, work on this principle, which was also made use of by the designers of the ENIAC. It is not now, however, much in favour for large decimal electronic machines, partly because of the inherent limitation on speed imposed by the existence of the intermediate states referred to, and partly because it requires the use of inefficient coding schemes. It still, however, has some advantages for use in small, and comparatively slow, machines based on the use of 10-state gas counter tubes.

In adding devices of the second kind, code groups representing the

digits to be added together are presented to static coding circuits which generate code groups representing the sum and carry digits of the result. These are formed, after an interval depending on the time constant of the circuit used, without the circuit passing through a sequence of distinct intermediate states. Circuits of this type are adders pure and simple, and need to have registers associated with them in order to form accumulators.

An adding circuit for use in a decimal calculating machine must be capable of accepting two decimal digits together with a carry digit

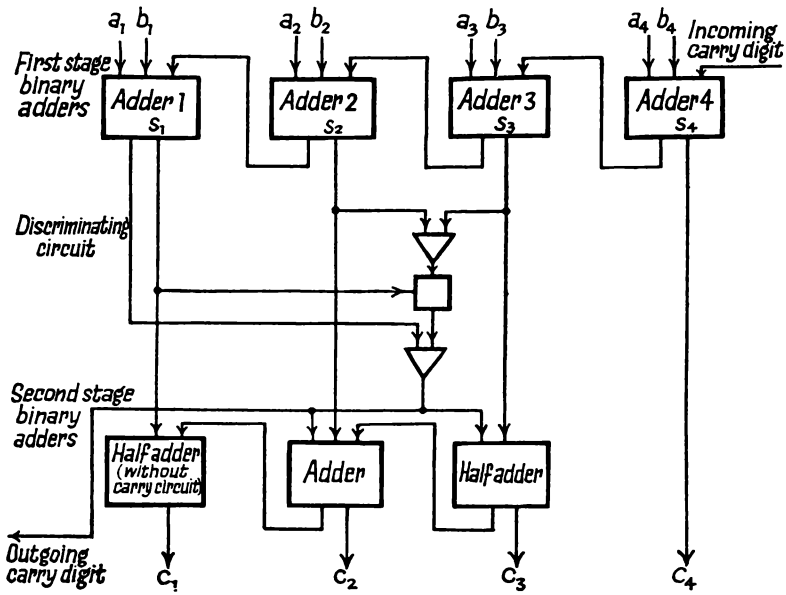


Fig. 6·34.—A binary-coded decimal adder.

from the previous stage, and of giving out a sum digit and a carry digit. It is thus a device with three inputs, two decimal and one binary, and two outputs, one decimal and one binary. It is also possible to construct two-input adders which can be designed to accept either two decimal inputs or one decimal and one binary input. In each case a decimal sum digit and a binary carry digit are generated. A pair of two-input adders, one of each type, may be connected to form a complete adder in rather the same way that a pair of binary half-adders may be connected to form a complete binary adder.

Comparatively simple adders based on binary adding circuits may be constructed for use with coding schemes, such as (*a*) and (*b*) in

Table 6·1, in which the binary code groups corresponding to the successive decimal digits are arranged in ascending numerical order according to their values as binary numbers. Such binary coding schemes are sometimes called *continuous*.

In a continuous binary coding scheme the decimal digit d is represented by $d+k$ expressed as a binary number; k is a constant such that $0 \leqslant k \leqslant 15$. The first stage in the addition of two digits d and d' expressed in this coding scheme is to add together the corresponding binary numbers $d+k$ and $d'+k$ by means of a binary adder. The

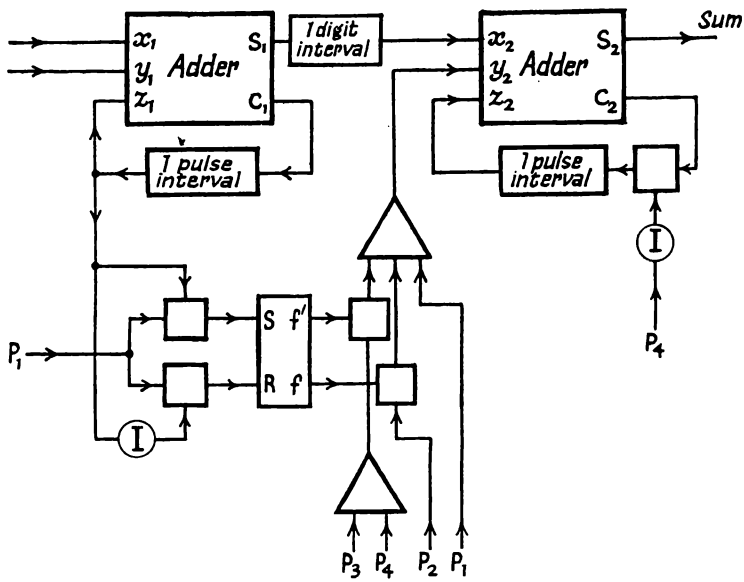


Fig. 6·35.—A serial decimal adder for the excess-three code. For the recurrent waveforms P_1, P_2, P_3, P_4 see Fig. 6·36.

result, $d+d'+2k$, must then be operated upon so as to produce a carry digit (either 0 or 1) and a sum digit expressed in the same coding scheme as the digits being added. Two cases arise. If $d+d'+2k < 10+2k$, the carry digit is 0 and the sum digit must be formed by subtracting k. If $d+d'+2k \geqslant 10+2k$, the carry digit is 1 and the sum digit must be formed by subtracting $10+k$ or, what is the same thing, by adding $6-k$ (modulo 16). Thus, if a continuous binary coding scheme is used, a decimal adding circuit can be formed with two binary adding circuits and a discriminating circuit. (For the present purpose the term 'adding circuit' is taken to cover circuits for subtraction as well as addition.) Since the second adding

circuit has only to add one of two fixed numbers to the output of the first adding circuit, it may be possible to use a simpler circuit than is required for complete binary addition.

Fig. 6·34 shows a decimal adding circuit designed for use with the coding scheme (*a*) of Table 6·1, for which $k=0$. a_1, a_2, a_3, a_4 and b_1, b_2, b_3, b_4 are the binary digits (in decreasing order of significance)

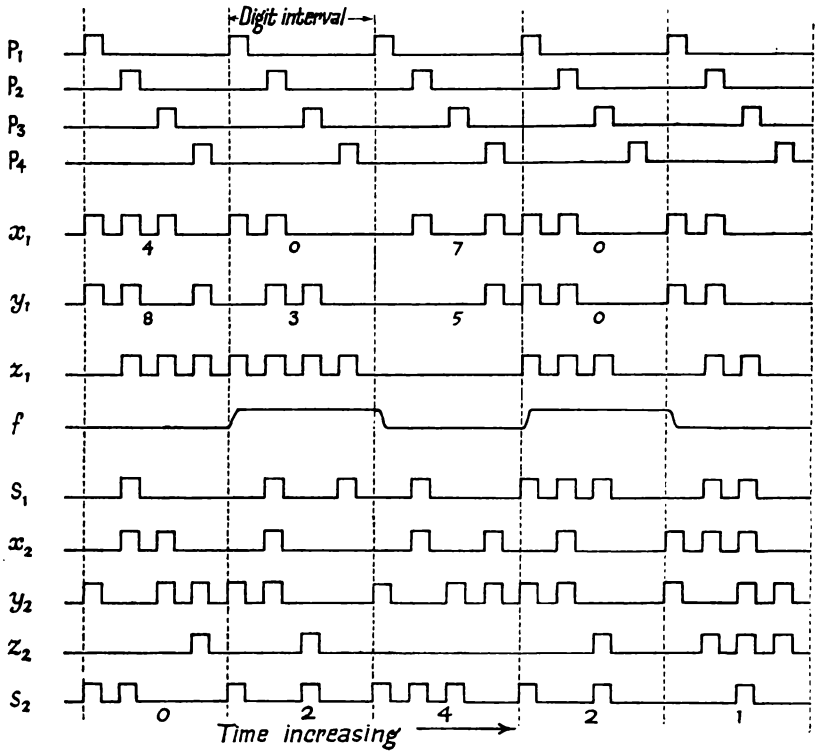


Fig. 6·36.—Recurrent waveforms required for the adder shown in Fig. 6·35 and waveforms illustrating the addition of the two numbers 704 and 538.

of the coded representations of the decimal digits being added together, and c_1, c_2, c_3, c_4 are the corresponding digits of the sum. For the first stage four three-input binary adders are used. The discriminating circuit shown is that derived on p. 228 modified to accept a fifth input variable. The second stage of the addition consists in adding either 0 or 6 (in binary form, 110) according as the output of the discriminator is 0 or 1. This can be accomplished with one adder and two half-adders; the left-hand half-adder does not require a carry output.

If a practical design for an adder working on this principle were required it would be worth while examining the discriminating circuit and the second-stage binary adders to see whether any economy in equipment could be effected by regarding them as a single unit, namely a five-input, five-output coder. Whether this would be possible would depend to some extent on the type of electronic circuit it was proposed to use and on the speed at which the adder was required to operate. Allowing for the discriminating circuit, it will be seen that the complete decimal adder requires about as much equipment as two four-stage binary adders.

The coding system with $k = 3$ (Stibitz's excess-three code) shown in column (*b*) of Table 6·1 is the only continuous code which gives complements by inversion. It also has the advantage that no separate discriminator is required, since the choice of the second-stage operation (which consists either in adding or subtracting 3) depends on whether or not there is a carry output from the most significant binary adder in the first stage. However, the operations of adding and subtracting 3 are such that relatively little economy is possible in the second stage adders, one half-adder, two complete adders, and one adder without carry being required. The excess-three code is very suitable for use in a purely serial decimal machine in which a single channel is used for the transmission of numbers. The logical diagram of an adder for use in such a machine is shown in Fig. 6·35 and waveforms for a typical addition in Fig. 6·36. It will be noticed that a set of recurrent digit pulses is assumed to be available, and the circuits required for generating these must be taken into account when assessing the total amount of equipment required.

A general approach to the problem of designing an adder, which may be used whatever the coding scheme, is to regard the adder as a coding device. This will be illustrated by discussing the design of an adder with two decimal inputs, suitable for use with the following two-out-of-five coding scheme based on the four-digit code given in column (*h*) of Table 6·1.

0	0 0 0 1 1
1	1 0 0 0 1
2	1 1 0 0 0
3	0 1 1 0 0
4	0 0 1 1 0
5	1 0 0 1 0
6	0 1 0 0 1
7	1 0 1 0 0
8	0 1 0 1 0
9	0 0 1 0 1

It will be assumed that the adder is to use diodes as switching elements. The example is taken from 'Synthesis of Electronic Computing and Control Circuits', p. 190 (Harvard Computation Laboratory).

The first step is to draw up an addition table in which the two inputs

and the sum output are expressed in the chosen coding scheme and in which the carry output is shown as a single binary digit. A section of this table is shown as Table 6·2.

TABLE 6·2

v_1	w_1	x_1	y_1	z_1	v_2	w_2	x_2	y_2	z_2	C	V	W	X	Y	Z
0	0	0	1	1	0	0	0	1	1	0	0	0	0	1	1
0	0	0	1	1	1	0	0	0	1	0	1	0	0	0	1
0	0	0	1	1	1	1	0	0	0	0	1	1	0	0	0
		.					.						.		
		.					.						.		
		.					.						.		
1	0	0	0	1	0	1	0	0	1	0	1	0	1	0	0
1	0	0	0	1	1	0	1	0	0	0	0	1	0	1	0
		.					.						.		
		.					.						.		
		.					.						.		
1	0	1	0	0	1	0	0	1	0	1	1	1	0	0	0
1	0	1	0	0	0	1	0	0	1	1	0	1	1	0	0
		.					.						.		
		.					.						.		
		.					.						.		

The binary digits representing the two inputs are denoted by v_1, w_1, x_1, y_1, z_1 and v_2, w_2, x_2, y_2, z_2, the sum output by V, W, X, Y, Z, and the carry output by C. By inspection of the addition table it is then possible to express V, W, X, Y, Z, and C as switching functions of the input variable. For example, the expression for V is

$$\begin{aligned} V = {} & y_1z_1(v_2z_2 + v_2w_2 + v_2y_2 + v_2x_2) \\ & + v_1z_1(y_2z_2 + v_2z_2 + x_2y_2 + w_2z_2) \\ & + v_1w_1(y_2z_2 + w_2x_2 + v_2y_2 + x_2z_2) \\ & + w_1x_1(v_2w_2 + x_2y_2 + w_2y_2 + x_2z_2) \\ & + x_1y_1(v_2z_2 + w_2x_2 + v_2x_2 + w_2y_2) \\ & + v_1y_1(y_2z_2 + v_2w_2 + w_2z_2 + v_2x_2) \\ & + w_1z_1(v_2z_2 + v_2y_2 + w_2z_2 + x_2z_2) \\ & + v_1x_1(y_2z_2 + x_2y_2 + v_2y_2 + w_2y_2) \\ & + w_1y_1(w_2x_2 + x_2y_2 + v_2x_2 + x_2z_2) \\ & + x_1z_1(v_2w_2 + w_2x_2 + w_2z_2 + w_2y_2). \end{aligned}$$

Note that it has not been necessary to include any dashed variables in this expression, since by the fundamental property of a two-out-of-five coding scheme we must have, for example,

$$v_1w_1x_1'y_1'z_1' = v_1w_1. \tag{17}$$

The expression for V may be simplified by making use of equations of the following type, which also follow from the fact that a two-out-of-five coding scheme is being used:

$$v_2 = v_2(w_2 + x_2 + y_2 + z_2).$$

It will be found that eight such simplifications may be made so that the final result for V is

$$\begin{aligned} V = {} & y_1z_1v_2 + v_1x_1y_2 + w_1y_1x_2 + x_1z_1w_2 \\ & + y_2z_2v_1 + v_2x_2y_1 + w_2y_2x_1 + x_2z_2w_1 \\ & + v_1z_1v_2z_2 + v_1z_1x_2y_2 + v_1z_1w_2z_2 \\ & + v_1w_1w_2x_2 + v_1w_1v_2y_2 \\ & + w_1x_1v_2w_2 + w_1x_1x_2y_2 \\ & + x_1y_1v_2z_2 + x_1y_1w_2x_2 \\ & + v_1y_1v_2w_2 + v_1y_1w_2z_2 \\ & + w_1z_1v_2z_2 + w_1z_1v_2y_2 + w_1z_1w_2z_2. \end{aligned} \tag{18}$$

A diode matrix based on this expression is shown in Fig. 6·37. As usual, the diodes are arranged in two levels so that not more than two are operated in series. If this restriction were not imposed, or if switching elements of a different type were to be used, it would be worth while considering whether further reduction of equation (18) would lead to greater economy in components.

For a complete adder five other matrices of similar type to that shown in Fig. 6·37 must be designed for generating X, Y, Z, W, and C. If these matrices are completely independent, full advantage will be taken of the self-checking property of the two-out-of-five code. If self-checking is not insisted on, some equipment may be saved by judicious combination of the matrices, although in view of the limitations of diodes when used as switching elements it is likely that extra amplifying or driving valves will be required.

In addition to being self-checking a two-out-of-five code has the further property that it enables diode matrices for generating arbitrary functions to be constructed without making use of the inverses of the input variables. That this is so may be seen by noting that all dashed variables can be eliminated from the switching function at the outset by making use of equation (17) and similar equations. The adding circuit just described consequently requires only 10 driving valves, whereas analogous circuits designed for use with a four-digit code would require 16.

There is no reason why the same coding scheme should be used in all parts of a machine, and in certain circumstances it may be worth while to design an adder, say, to accept inputs expressed in a two-out-of-five code and to generate the output in a four-digit code. An example of such an adder can be obtained by omitting the matrix for generating V from the adder just described; the output would then be in the code given in column (*h*) of Table 6·1.

The code given in Table 6·2 is a particularly advantageous one as regards the construction of adding circuits. If the reader cares to attempt the design of an adding circuit for use with, for example, a five-digit code based on column (*f*) of Table 6·1 he will find that the

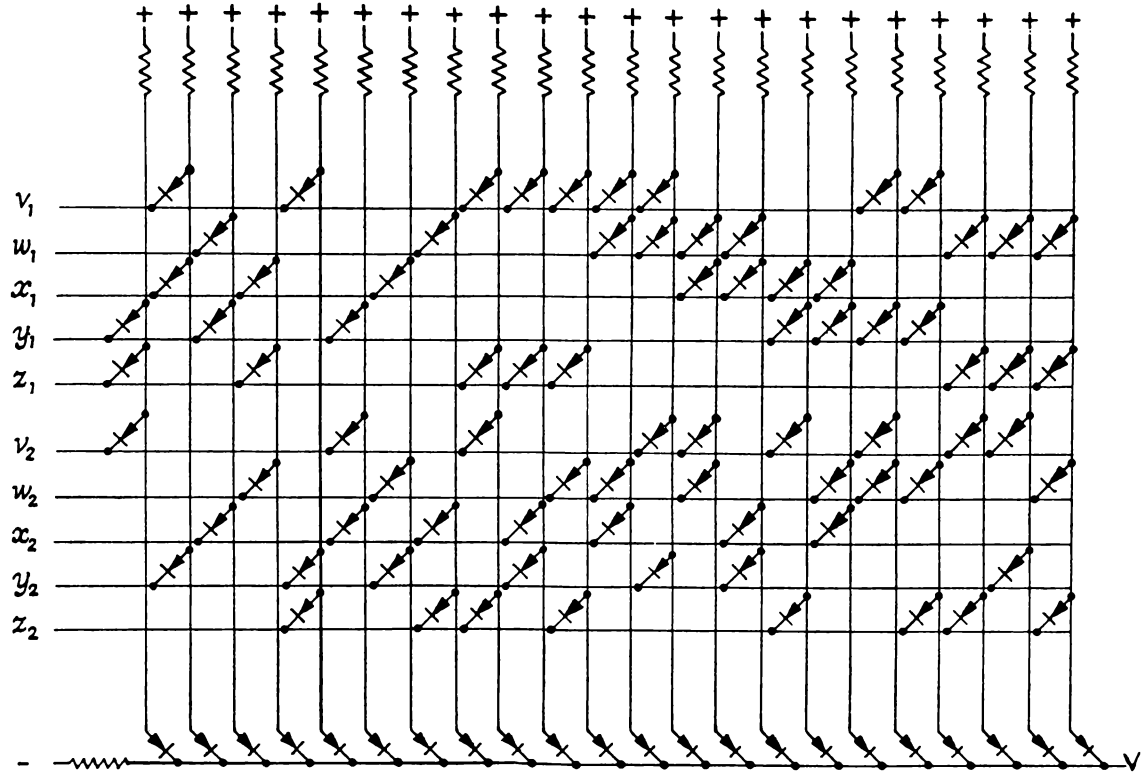


Fig. 6·37.—Diode matrix for use in a decimal adder working with a two-out-of-five code.

Boolean switching functions will not simplify to anything like the same degree. This will impress on him the fact that the amount of equipment in coded decimal computing circuits depends very markedly on the code used.

Decimal Multipliers

No more than a brief account of the various ways in which multipliers may be constructed for decimal calculating machines will be attempted in this section. The reader who wishes to follow the matter further is advised to study the accounts of various decimal machines which have been published. No new principles are involved, but there is scope for the exercise of much ingenuity within the framework of the technique employed for a particular machine and, as in other parts of a computer, there is a balance to be struck between simplicity of equipment and speed of operation.

The simplest method of multiplication is by repeated addition. The digits of the multiplier are examined one by one and an appropriate number of multiples of the multiplicand added into the accumulator. The process is repeated, after suitable shifting operations have been performed, for each digit of the multiplier. The need to handle negative numbers introduces complications very similar to those encountered in designing a binary multiplier.

The multiplication can be speeded by a factor of about two if 'short-cutting' is used. This procedure will be familiar to anyone who has used a desk calculating machine in which multiplication is not performed automatically. Suppose, as an example, that the multiplier is 17; the multiplicand is first added into the accumulator twice, the number in the accumulator shifted one decimal place to the left and the multiplicand then *subtracted* three times. This gives the correct result and involves only five additions or subtractions compared with eight, which is the number required if short cutting is not used.

Another method of reducing the average length of time taken to perform a multiplication is to arrange that, before the building up of the product begins, the multiples from 2 to 9 of the multiplicand are formed and stored in a set of registers set aside for that purpose. As each digit of the multiplier is examined it is then only necessary to perform one addition, namely, the adding in of the correct multiple. The number of multiples which must be formed, and the number of registers which must be set aside for holding them, may be reduced from eight to four if short-cutting is used.

Further economy in the use of registers is possible if static coding circuits are used for generating some of the multiples. This is most conveniently done for the second and fifth multiples, since in these cases, as may easily be verified, the nth digit of the multiple depends only on the nth and $(n-1)$th digit of the original number.

Another method of obtaining fast multiplication is to make use of a built-in multiplication table. This device was first used (in a desk computing machine) in about 1890. The product of two decimal digits is a two-digit number; two separate generators are therefore necessary for generating what will be called, respectively, the right-hand and left-hand components of the product. As an example, suppose that the multiplicand is 27915 and that the multiplier digit under consideration is 7; then the quantities produced by the right-hand and left-hand component generators respectively are 49375 and 146030. In the ENIAC, which uses this method of multiplication, the right-hand and left-hand components corresponding to the various multiplier digits taken in turn are accumulated in separate accumulators and added together at the end. An alternative method would be to provide a special adding circuit to add the right-hand and left-hand components together as soon as they are formed, and then to add the sum into the accumulator. This means that the arithmetic unit must contain two adders instead of one, but the addition of the two partial sums at the end is avoided. A further possibility is to use an adder which will accept three decimal inputs so that the right-hand and left-hand components may be added simultaneously to the number standing in the accumulator. The adding circuits used in the individual stages of such an adder would need to be capable of accepting three decimal inputs, together with a carry input, which could be either 0, 1, or 2, and of giving one decimal output and a carry output, which could also be 0, 1, or 2.

A further and elegant method by which multiplication may be carried out depends on the use of static coding circuits for multiplication and division by two. The method may best be regarded as multiplication in the binary scale, although the numbers concerned are expressed in the decimal scale and a decimal adder is used. A whole number expressed in decimal notation may be converted to binary form in the following manner. The number is repeatedly divided by two, any remainder being neglected. Before each division the number is examined to determine whether it is odd or even; if it is odd a 1 is recorded, and if it is even a 0 is recorded. The 1s and 0s obtained build up the binary form of the number, beginning with the least significant digit. In the form of multiplication based on this process, the multiplier is repeatedly halved (remainders being neglected) and the multiplicand is repeatedly doubled. Whenever the number resulting from halving the multiplier is odd, the number resulting from doubling the multiplicand is added into the accumulator before it is doubled again. This process continues until the multiplier is reduced to 0, at which time the accumulator contains the product.

ERROR-DETECTING AND ERROR-CORRECTING CODES

The reader will already be familiar with the use of special codes to give protection against accidental error. Examples are the two-out-of-five code and the bi-quinary code for representing decimal digits. A particularly useful device is the association of an extra 'parity' digit with each binary number. The parity digit is chosen so that the sum of all the digits (including the parity digit) is either zero (modulo 2), in which case the number is said to have even parity, or unity (modulo 2), in which case the number is said to have odd parity. The occurrence of a single error during transmission may then be detected by observing that the parity of the number has changed.

If a parity digit is used only half the combinations which can be transmitted are valid in the sense that they represent admissible numbers. In order to convert one valid combination into another valid combination, it is necessary to replace *two* digits by their inverses. If, as a result of an error in transmission, one such interchange only takes place, an invalid combination results and the error is revealed. If a larger proportion of invalid combinations is permitted the code may be made such that errors can not only be detected but can also be corrected. This is possible if at least three inversions of digits are necessary in order to change one valid combination into another valid combination. With such a code, if a single inversion takes place, the resulting combination differs from the original combination in one digit only, while it differs from all other valid combinations in at least two digits. Simple inspection, therefore, makes it possible to correct the error.

An example of an error-correcting code in which four inversions at least are necessary in order to pass from one valid combination to another is given in Table 6·3. The number to be transmitted has 40

TABLE 6·3

1	0	1	0	0	1	1	1	0	1	*0*
0	1	1	0	1	0	1	1	1	1	*1*
0	0	0	1	0	1	1	0	1	1	*1*
1	1	1	1	1	0	1	0	0	0	*0*
0	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*0*

digits and these are written in 4 rows of 10 digits each. Parity digits—shown in italic type—are added to each row and column, so that in the resulting 11 by 5 matrix the sum, modulo 2, of each row and column is zero. Thus, a single error occurring during transmission will be detected if the parity of each row and column is checked, and may be corrected by inverting the digit standing at the intersection of the row and column whose parities are found to be incorrect. When applied to the transmission of a single number in the manner just

described, this error-correcting code is rather uneconomic as regards the number of extra digits which must be transmitted for checking purposes. It can, however, be used economically when a block of, say, 40 numbers, each containing 40 digits, is being transmitted; in this case 81 redundant digits are required to give error-correction for 1,600 digits.

A systematic method of constructing efficient error-correcting codes has been given by Hamming. The construction of a seven-digit code of this type will be illustrated with the aid of Table 6·4. Of the seven digits $a_1, a_2, \ldots, a_7$, four may be chosen arbitrarily, and these can be conveniently taken to be the digits a_3, a_5, a_6, a_7, which are marked with a cross in Table 6·4. The remaining three digits, a_1, a_2, a_4, are then chosen so that the following relations are satisfied:

$$0 \equiv a_4 + a_5 + a_6 + a_7 \pmod 2$$
$$0 \equiv a_2 + a_3 + a_6 + a_7 \pmod 2$$
$$0 \equiv a_1 + a_3 + a_5 + a_7 \pmod 2.$$

The terms which occur in these three equations are indicated in Table 6·4 by stars arranged in three columns, each column corresponding to one equation. It will be noted that the stars are arranged systematically, in such a manner that if stars are replaced by 1s and blanks by 0s the binary numbers from 1 to 7 are obtained.

TABLE 6·4

a_1				*
a_2			*	
a_3	×		*	*
a_4		*		
a_5	×	*		*
a_6	×	*	*	
a_7	×	*	*	*

After transmission a check on the accuracy of the received digits may be obtained by forming the quantities l, m, n, given by

$$l \equiv a_4 + a_5 + a_6 + a_7 \pmod 2$$
$$m \equiv a_2 + a_3 + a_6 + a_7 \pmod 2$$
$$n \equiv a_1 + a_3 + a_5 + a_7 \pmod 2.$$

If no error of transmission has occurred l, m, n will all be zero. If one of the digits has become inverted, one or more of these three quantities will be equal to unity, and it will be possible to deduce from Table 6·4 which is the inverted digit; for example, if $l = 1$, $m = 0$, $n = 1$, then a_5 must be in error, since an error in no other digit could give rise to this combination of values of l, m, n. It will be found that if l, m, n, in that order, are taken to be the digits of a binary number equal, say, to p, then the pth digit in the transmitted number is the

erroneous one; for example, if $l=1$, $m=0$, $n=1$, then $p=101$, so that the fifth digit, namely a_5, is in error. This pleasing feature of the code is a consequence of the systematic arrangement of the stars in Table 6·4 to which attention has already been called. The code just described has no more redundancy than is necessary to provide error-correction against a single error. The addition of another digit, a_8, chosen so that the parity of the whole number shall be even, converts the code into one which is error-correcting against a single error and error-detecting against two errors. If still more security against error is required the addition of further digits is necessary.

The scheme shown in Table 6·4 can readily be extended to yield codes containing a larger number of digits. For example, if another column of stars is added in Table 6·4, a 15-digit code may be obtained in which 11 digits may be chosen arbitrarily and four are redundant. More generally, a code with $2^n - 1$ digits, n being redundant, may be constructed by this method.

Checking by Weighted Counts

A well-known method of checking the results of arithmetic operations is that of 'casting out the nines'. To apply this method to check the sum of two numbers x and y, the sum (modulo 9) of the digits of x and the sum (modulo 9) of the digits of y are first found. If these quantities are denoted by x_c and y_c, then

$$x_c + y_c \equiv \text{sum of digits of } x+y \pmod 9.$$

This may be proved by noting that the contribution of a particular digit (say the nth from the least significant end of the number) to x is $10^{n-1}d$, where d is the value of the digit, and that the contribution of the same digit to $x - x_c$ is $(10^{n-1} - 1)d$, which is either zero or divisible by 9. It follows, when the contributions of all digits are taken into account, that

$$x - x_c \equiv 0 \pmod 9.$$

It similarly follows that

$$y - y_c \equiv 0 \pmod 9,$$

and also that

$$(x+y) - (x+y)_c \equiv 0 \pmod 9.$$

From these equations the result stated above may be readily deduced.

The method of checking just described may be used with numbers expressed in any base r, where $r > 2$; all sums must then be calculated modulo $r - 1$. The method fails, however, if $r=2$, that is, if numbers are expressed in the binary scale. The sum of the digits, modulo 2, of a binary number gives the parity of the number, but it is, in fact, not

possible to devise a method of checking arithmetic operations in the binary scale by using a single parity digit only. This difficulty may be overcome by grouping the binary digits and regarding the numbers as being expressed in some scale s, where s is a power of 2, say, 2^{α}, where α is an integer. The above method may then be applied using the sum of the digits of each number calculated for the scale of s. This sum may be found directly by forming the weighted count $\sum w_i d_i \pmod{s-1}$ of the digits d_i of the original binary number (d_0 being the least significant digit), where the weights w_i are given by

$$w_i = 2^j, \text{ where } j \equiv i \pmod{\alpha}.$$

If $s = 8$, for example, so that $\alpha = 3$, this weighted count is

$$d_0 + 2d_1 + 4d_2 + d_3 + 2d_4 + 4d_5 + d_6 + \ldots \pmod 7.$$

The sums modulo $s - 1$ may be formed in an accumulator fitted with end-around carry (see p. 41).

In the RAYDAC, a computer built by the Raytheon Corporation, weighted counts were not only used to check the accuracy with which arithmetic operations were performed but were also transmitted along with numbers and used to check the accuracy of transmission. A better check on the accuracy of transmission can be obtained in this way than by the use of a single parity digit, although in most circumstances the latter is sufficient.

Cyclic or Reflected Codes

The following binary code for representing the natural integers can be extended without limit:

0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	1	
3	0	0	1	0	
4	0	1	1	0	
5	0	1	1	1	
6	0	1	0	1	
7	0	1	0	0	
8	1	1	0	0	&c.

This code has the property that, in order to pass from one integer to the next higher integer, it is necessary to change one binary digit only. It has been found useful in connexion with the design of commutators for giving a digital indication of the position of a shaft, since a small movement of the shaft can never cause more than one digit to change, whereas, if simple binary representation is used, a small movement can cause several digits to change (e.g., from 01111 to 10000); in the latter case it would be difficult to design the commutator so that all digits changed at exactly the same instant, and if they did not do this

a large momentary error could exist in the transmitted binary representation of the shaft position.

A decimal code with a similar property is as follows:

0	0	10	19	20	20	30	39	
1	1	11	18	21	21	31	38	
2	2	12	17	22	22	32	37	
3	3	13	16	23	23	33	36	
4	4	14	15	24	24	34	35	
5	5	15	14	25	25	35	34	
6	6	16	13	26	26	36	33	
7	7	17	12	27	27	37	32	
8	8	18	11	28	28	38	31	
9	9	19	10	29	29	39	30	&c.

Such codes are known as *cyclic* or *reflected*, and have a number of interesting properties. For example, a number x expressed in binary notation can be converted to the reflected binary code by forming the sum of x and $2x$ without carry (i.e., taking the sum, modulo 2, of corresponding digits) and dropping the right-hand digit; e.g., if $x = 110$, we have

$$\begin{array}{r} 1\ 1\ 0 \\ 1\ 1\ 0\ 0 \\ \hline 1\ 0\ 1\ (0). \end{array}$$

A number expressed in reflected binary code can be converted to ordinary radix form (in any scale) by assigning the following weights to the digits, beginning with the least significant, 1, 3, 7, 15, 31, &c., and prefixing alternate signs; e.g., 10011001 represents $255 - 31 + 15 - 1$, which is equal to 238.

CHAPTER 7

THE DESIGN AND OPERATION OF DIGITAL COMPUTERS

THIS chapter will contain a general discussion of the various factors which must be taken into account when deciding on the logical design of a digital computer. It will also contain some discussion and evaluation of the more elaborate programming techniques which may be used, and of the problems encountered in organizing a computing centre. These subjects are all closely related. On the one hand, it is the job of the machine designer to provide the programmer with a machine which is capable of efficient and convenient use, and, on the other hand, technical developments in programming have an influence on the design of machines.

Many different approaches may be made to the subject of logical design according to the stress which is put on the various desirable features that a machine may have. Speed of operation, reliability, ease of maintenance, ease of programming, maximum protection against intermittent error, ruggedness of mechanical design, fast input and output, large storage capacity—these are all features that the user of a machine would like it to have; they are not all mutually consistent, and there is room for divergence of opinion as to the means which should be taken to secure some of them. Again, the requirements that a machine must satisfy differ according to whether it is to be used for general scientific work, for statistical work, or for commercial work.

In so far as this book is concerned at all with engineering, it deals with the logical design of computers rather than with the detailed design of circuits. For the purposes of this chapter, however, a few remarks are necessary about the way the subject appears to an engineer charged with the overall responsibility for the design and construction of a machine.

THE DESIGN OF A DIGITAL COMPUTER

An engineer fresh to computing machine design will probably have had experience in some other electronic field. He will find himself most at home if his experience has been in radar or television, or in some other field in which short pulses are used. He will then be used to thinking in terms of pulse shapes and time intervals, rather than in terms of Fourier components and phases, as comes more naturally to an engineer trained in the design of communications equipment.

However, he will find at once the important difference that in computing circuits there are often two significant levels of signal only; the circuits are, therefore, frequently designed to be non-linear in their action so as to limit the signal amplitude to a maximum value on the one hand and to remove any small unwanted signals on the other. Thus, in computing machines, non-linear amplifiers are used more often than linear amplifiers.

Perhaps the most striking thing about the more ambitious electronic computing machines, compared with other electronic installations, is their scale. While it may not be unusual in other fields to find four or five thousand valves in a single installation, it is unusual to find this number of valves, forming part of a single device, which must all be working together. If one part of a computing machine goes wrong, the machine is, with few exceptions, useless, whereas in a telephone repeater station, for example, one channel can go out of action while the rest go on working. Moreover, one amplifier can be switched out of circuit and a spare amplifier switched in without more than a minor inconvenience arising from the temporary interruption of the service during switching. In a similar way it does not matter in television if interference, or a faulty connexion, causes an occasional frame of the picture to be imperfectly reproduced, since the amount of information being transmitted is greatly in excess of the minimum necessary, and a person viewing the picture can disregard a temporary flash. In a computing machine, on the other hand, a similar occurrence would mean that some section, large or small, of the work would have to be repeated; every pulse counts, and no irregularities can be tolerated.

The apparent simplicity of computing circuits in which two significant levels only are used has often led to over-optimistic estimates being made of the time required to build and put into operation a large computing machine; people tend to forget that, although the circuits may be simple, there are a very large number of them in a machine. This would not matter if machines consisted of a small number of different types of unit repeated again and again, all units of the same type working under identical conditions. This is not true of a majority of the electronic computing machines which have so far been completed, although it would seem to be an obvious direction in which design should develop, and more will be said about it later. It is important that the units should work under identical conditions; in a sense all machines consist of gates, flip-flops, and similar circuits, but usually in different parts of the machine the pulse amplitudes and shapes are different, and different standards of performance—for example as regards the attenuation of signals or the permissible level of breakthrough of unwanted signals—may be necessary.

In television or radar the circuits can be designed on the assump-

tion that the waveforms they are to handle are recurrent, and it does not matter if information is lost when an abrupt change occurs, for example, when switching from one camera to another in a television programme. In a computing machine, however, the correct handling of transients is important. In a delay store, for instance, for a long period of time there may be no pulses at all circulating—corresponding to all the digits in the store being 0—and then suddenly a train of pulses, corresponding to a block of 1s, may be transferred in. It would not do if the first few pulses entering the store were lost as a result of transients generated in the circuits. Thus it is necessary, with very few exceptions, that the circuits in an electronic computing machine should be designed to handle waveforms which are not even approximately recurrent.

Margin of Operation

It goes without saying that the first requirement for reliable operation of a computing machine is that the individual circuits should be so designed as to have an adequate margin of operation. This means that the voltage swings must be more than sufficient to provide good switching action and to allow for drift in the values of components and in the characteristics of valves. For a given type of circuit, however, it gets increasingly difficult to design for adequate margins as the speed of operation is increased, and the point is ultimately reached at which the circuit cannot be made to function reliably. If a higher speed still is insisted on, it is necessary to devise an improved form of circuit—probably one using more valves or consuming more power. There will be an upper limit to the speed at which this new circuit can be made to function satisfactorily, and eventually a point will be reached at which the circuit designer's ingenuity is exhausted and further increases of speed cannot be obtained with the circuit components available, unless, perhaps, at the expense of a prohibitively large power consumption. Operating margin is thus intimately connected with speed of operation. Frequently the computing engineer sets himself the task of getting the maximum possible speed out of circuits of a given type rather than that of designing a machine to work at a given speed. Provided that the speed is not pushed to the point at which reliability suffers, this is a praiseworthy aim, since there is no point in building a slow machine when a faster one can be built with the same quantity of equipment. It means that, in the long run, advances in circuit design are more likely to lead to higher overall speeds of operation than to machines in which the circuits have improved operating margins. Naturally much depends on the designer's judgement, and it is probable that in the past many designers have succumbed to the temptation of trying to make their machines faster than they should have done. It is particularly important to

bear in mind that, for the machine as a whole to have a satisfactory operating margin, the individual components must have much larger operating margins than if they were being used in isolation.

A reduction in speed beyond a certain point does not necessarily give greater reliability. Even if greater reliability can be gained in this way, it does not always lead to a greater output of work from the machine. For example, a machine which works for 60 per cent of the time will do more work in the long run than a machine running at half the speed which works for 95 per cent of the time. It is assumed that in these figures allowance is made for the time lost in re-running work in which errors have occurred through the malfunctioning of the machine. However, the routine operation of a machine which only works for 60 per cent of the time presents awkward administrative difficulties and places a great strain on the maintenance and operating staff. Some reduction in speed—say by a factor of 2—may therefore be regarded as a reasonable price to pay for a substantial increase in reliability, even though the overall output of the machine may be slightly reduced; but very slow machines, however reliable, are usually uneconomic.

Apart from the margin of operation of the individual circuits, the frequency of breakdown of the machine depends directly on the amount of equipment it contains. Thus any move to reduce the size of a machine by radical improvements in logical design is likely to lead to increased reliability. It is not always wise, however, to prune the individual circuits so that they contain the absolute minimum number of valves; it is noticeable that, when unsatisfactory circuits in a particular machine are replaced by circuits of improved design, the new circuits nearly always contain more valves than the ones they replace.

Complexity of Logical Design

The proportion of time that a machine is out of action depends not only on the frequency of occurrence of faults but also on the ease with which the faults can be located and rectified. This, in turn, depends on the complexity of the logical design. Consider two machines. The first consists of a large number of units of many different types, cross-connected in such a way that the correct functioning of each is dependent on the correct functioning of others. The second machine consists of a few main sections, each of which is composed of units of a small number of standard types, each type being repeated many times. The sections are functionally self-contained and there are relatively few connexions between them.

The second machine has a number of advantages over the first. In the first place the original design is likely to be of a higher standard, since the available design effort can be concentrated on a few different

types of unit rather than being dispersed over a large number of different types. Moreover, weaknesses in the design are more likely to be detected and eradicated during the production and testing of a batch of identical units, than when units are produced on an individual basis. For similar reasons it is easier to train maintenance staff, and their average efficiency will be higher. Again, a machine of simple logical design, with many identical units working in parallel channels, lends itself to rapid localization of faulty units, both by conventional methods and by the use of fault-locating programmes. Since there are only a few types of unit, it is practicable to provide an adequate number of spares which can be plugged in to replace units which are suspected of being faulty. Many faults can be cleared in this way, leaving the examination and repair of the defective unit to be carried out at leisure while the machine is operating.

On the other hand, the simplification of logical design in the second machine may have been achieved at the expense of making the machine larger than it would otherwise have been. If it were to contain three or four times as much equipment as the first machine, it might well be that the fact that there was more to go wrong would outweigh the advantages listed above. If the quantity of equipment in the second machine were only, say, one and a half times that in the first machine, there would almost certainly be a distinct gain.

So far cost has not been mentioned as a factor influencing design. This is because the money expended on early machines went mainly into development work, and the actual cost of the components used to build a machine in its final form and the cost of construction were not major parts of the total expenditure. This situation is changing now that machines are beginning to be produced on a production line. Some of the arguments in favour of a machine with simple logic and repeated units carry less weight under these conditions. In the long run, if the design of computers reaches stability—a thing which is not likely to happen for a considerable time—the machine which has the least amount of equipment in it is likely to prove the most economical to purchase and to maintain, even though its logical design is not as straightforward as that of rival machines. This presupposes that the machine has survived a fairly long period of field trials, during which all weaknesses in the design have been eliminated. Complex logical design will always be a disadvantage from the maintenance point of view, but less so if a large body of maintenance experience with the same type of machine has been built up.

Error-detecting Circuits

The advisability of including in a machine devices for detecting errors has been discussed earlier in this book. From the point of view of the programmer such devices are a luxury in that it is quite possible

for him to get on without them; moreover, if their inclusion substantially increases the complexity of the machine, and makes it thereby more liable to break down, they may be more trouble than they are worth. In the early days of computing machine design this latter consideration led many designers (though not all) to look on error-detecting devices with disfavour. Every technical advance, however, which leads to more reliable operation of large machines reduces the force of this argument, and there is perhaps now a general swing of opinion in favour of error-detecting devices. On the other hand, further steady progress towards extreme reliability may counteract this tendency by making them less necessary.

The need for automatic error detection is most felt when large quantities of information are being transmitted from one place to another. In a computer large quantities of information are handled by the input and output mechanisms and by the access circuits of the main and auxiliary stores. Many designers now would favour some form of redundancy check at these points. Particularly useful is the storage of a redundant parity digit with each word when it is held in the main store. In a machine with a 40-digit word length, to do this makes the store larger by one-fortieth than it would otherwise be, and, in addition, necessitates the provision of circuits for generating the parity digit when writing, and for checking the parity when reading.

The automatic checking of arithmetic operations is more difficult, and complicates considerably the arithmetic unit. Fortunately it seems to be a general experience with many different types of machine that the arithmetic unit gives rise to comparatively few errors; moreover, since the maximum number of significantly different states which it can assume is relatively small, it is possible to test it adequately by means of test programmes. There would seem, therefore, to be a case for omitting any form of automatic error detection in the arithmetic unit, even if it is included elsewhere.

Although programmers would vote for the inclusion of automatic error-detecting devices—provided they could be had without compensating disadvantages—it is by regarding them as items of built-in test gear that the strongest case can be made out in their favour. They greatly facilitate the maintenance of a machine in a high state of efficiency by drawing attention to trouble as soon as it occurs and by giving an indication of the section of the machine concerned. Without help of this kind the location of a rarely occurring fault—even establishing that the fault exists—can take a long time; it is often surprisingly difficult to decide whether a fault is occurring in the store, in the transfer circuits, or in the arithmetic unit.

Whenever possible, error-detecting circuits should be designed so that the machine can be run without them if they become faulty, or

for purposes of test. If this can be done, then the objection that the inclusion of error-detecting circuits increases the overall fault time of a machine is to some extent met. It may even be possible to arrange that a part of the store normally used to hold a parity digit can be temporarily switched in to replace some other part of the store which may be giving trouble.

The designer has a choice of various actions which may be initiated when an error occurs and is automatically detected. The simplest is for the machine to stop. This, however, is frequently unhelpful. If the error is a trivial one—for example, the incorrect printing of a single character which happens not to be vital—it would be far better for the machine to continue with the programme, having printed some indication that the error has occurred. On the other hand, if the error is more serious it is better that the machine should repeat the section of the calculation affected. These varying requirements can be met if an error indication causes control to be sent to some location in the store, preferably one specified by the programmer. The programmer can then arrange that the detection of an error is followed by whatever action he considers most suitable, whether it be for the machine to stop, to print an indicating symbol, or to repeat a section of the calculation.

Built-in error-detecting circuits are most useful in machines intended for data handling rather than for mathematical calculation. The designers of the UNIVAC were pioneers of a completely checked electronic computer, and it is significant that their machine was, from the beginning, intended for business calculations, census reductions, and similar purposes.

FAULTS

Faults which are due to the sudden and complete failure of a valve or a component are not likely to go undetected for long, and it is usually easy to locate the faulty valve or component. If such faults were all one had to contend with, there would not be much difficulty in maintaining a satisfactory standard of serviceability in an electronic computer. Valves are much more likely to fail than components, but it is impossible to give precise figures for the failure rate to be expected, mainly because the amount of information relating to the performance of valves under the conditions encountered in computers, and capable of unambiguous interpretation, is not great. Such information as there is tends to show that valves vary greatly as regards liability to sudden failure, both from type to type and, within the same type, from maker to maker, or even from one production run to another. Generally speaking, however, given a machine containing 5,000 valves, one would not expect catastrophic failures to occur much more frequently than once a week. It seems to be generally

agreed that the chance of failure occurring in a given valve during the first few hundred hours of its life is relatively high, and that, thereafter, the chance remains roughly constant, even though the valve may survive for many thousands or even tens of thousands of hours. There does not seem to be any way of predicting when a catastrophic failure is about to occur, and as far as this type of failure is concerned it is not sound policy to replace valves simply because they have reached a certain life.

More numerous—and more troublesome to locate—are faults due to changes in the characteristics of valves, or in the values of components. Such changes are always going on in any electronic computer, but it is only when the total change reaches a sufficient magnitude that trouble shows itself. Very likely the fault, when it occurs, will be of a programme-sensitive nature, that is, the machine will work perfectly on most programmes, but will give trouble on one or two particular programmes, although there is usually nothing to indicate why those programmes should be singled out in that way. All electronic computers, as far as I am aware, have at some time or other exhibited programme-sensitive faults; and all their designers—if they have not had previous experience of the day-to-day running of a large computer—have been surprised that this should be so. The occurrence of programme-sensitive faults adds quite appreciably to the troubles of the organizers of a computing centre, as well as to those of the maintenance staff.

Programme-sensitive faults can be caused in many ways. An obvious example is by the failure, or partial failure, of a D.C. restoring diode; this causes the coupling circuit of which the diode forms part to develop a 'memory', so that pulses occurring at the ends of long trains are treated differently from pulses occurring at the beginnings of trains, or in isolation. Programme-sensitive faults can also occur if a change in the duty cycle in one part of a machine can cause a surge which affects another part via the power-supply equipment. Sometimes parasitic oscillations are generated in certain circuits when the momentary condition of the machine is favourable to their growth. Programme-sensitive faults due to the latter cause, and indeed programme-sensitive faults generally, can be quite difficult to trace; not infrequently, even when the fault has been cured by making an adjustment, or by replacing a valve in a particular part of the machine, the mechanism of the fault is still far from clear. Very rarely can any difference be seen in the patterns displayed on an oscilloscope, connected to the suspected circuit, before and after the adjustment is made.

Most early machines suffered severely from the occurrence of programme-sensitive faults, but as the causes of such faults become better understood it is to be expected that engineers will be able to design

machines which are less prone to them. It is a subject, however, on which it is difficult to do experimental work, since, by their nature, programme-sensitive faults occur only in large digital computers operating under normal working conditions, and cannot be simulated in test set-ups using a small amount of equipment and operating on fixed programmes.

When the possibility of the occurrence of programme-sensitive faults exists, it is not possible to give a straightforward answer to the question whether the machine is completely free from faults. The best that can be said is that the machine has satisfactorily passed a given sequence of tests and has performed a certain number of programmes correctly; there can be no guarantee, however, that some particular programme will not be found to give trouble, even though the other programmes will still go through without error. The fact that the machine is liable to develop programme-sensitive faults means that the testing period must be much longer than would otherwise be necessary, in order that the machine may be tested on a sufficient variety of test programmes to give a good chance that it will not make errors during operation.

Test programmes usually consist of two kinds. First, there are straightforward programmes designed to test every function of the machine, and secondly, there are test programmes derived from operational programmes which have proved particularly troublesome. It is well that the latter should not be kept the same for too long, since the maintenance staff will inevitably tend to adjust the machine so that it does the test problems which are current at any time—it has been remarked that a machine appears to 'learn' to do its test problems.

Marginal Checking

Fortunately there is available a system whereby the drift in component values and valve characteristics may, to a large extent, be brought under control. This is the system of *marginal checking* originally developed for the Whirlwind at M.I.T. The idea is that, in addition to being tested under normal working conditions, the machine should be tested with the circuits working under artificially induced conditions which give a smaller operating margin. For example, the machine may be tested with all the flip-flops in a certain section thrown into an unbalanced condition by an alteration in the bias voltage applied to one grid, it may be tested with attenuators inserted in certain of the pulse leads, or it may be tested with a reduced heater voltage. If the changes in applied voltages, or in circuit parameters, are suitably chosen and systematically applied, the effects of drift will show up as faults under marginal conditions before the drift has gone far enough to cause faults under operational conditions.

Marginal checking, therefore, enables a large number of the faults that would otherwise occur during operating hours to be forestalled, and the remedies to be applied during periods of scheduled maintenance.

Generally speaking, it is desirable that each circuit should be tested in two marginal conditions, one on each side of the normal operating point. The amount by which the operating conditions of the circuits are changed during marginal checking is a matter for experiment; usually this is fixed, but some engineers have found it desirable to make it continuously variable and to measure and keep a record of how much variation can be tolerated before errors occur. In some circumstances marginal checking may be conveniently performed by injecting an alternating voltage of mains frequency in series with one or more of the power supplies. The machine can then, in one operation, be tested both at the normal operating voltage and over a range of voltages on either side.

There is no doubt that the provision of a good system of marginal checking can improve the serviceability of a machine appreciably. However, it must not be thought that any system of marginal checking will work wonders in this respect. Not only must the system be carefully planned in the first place to be as comprehensive as possible, but it must also be tested under operational conditions to make sure that it really does what it is intended to do. Of two systems tried on the EDSAC one was found to lead to a distinct improvement in serviceability while the other was disappointing, although on the face of it both systems initially appeared promising.

Quite apart from its use in forestalling faults which would otherwise occur during operating hours, marginal checking is of great assistance to the maintenance staff in locating elusive faults. For this purpose it is necessary to be able to apply marginal conditions to individual circuits in the machine. It is often possible to locate a circuit which is giving trouble by noting whether there is any change, for better or worse, in the symptoms exhibited by the machine when marginal conditions are applied to the various circuits in turn. Sometimes a temporary cure may be effected by putting a particular circuit into what is normally a marginal condition and leaving it in that condition during operation. This is sometimes useful when a machine is operated in the absence of maintenance staff.

Intermittent Faults

An important category of faults not so far considered consists of those which are of an intermittent nature. These may be due to causes inside or outside the machine. Fortunately most of the circuits used in computers are inherently insensitive to interference generated by electrical machinery; electrostatic storage systems incorporating

high-gain amplifiers are exceptions, but even here the effects of interference may be reduced to negligible proportions by adequate screening and filtering of power supplies.

All machines will make mistakes if the power supplies are subject to short-term irregularities which cannot be smoothed out by voltage-regulating circuits. In particular, some public alternating-current supply mains are subject to short interruptions, lasting a few cycles, which occur when transformers are switched in and out of circuit or voltage tappings adjusted. Such interruptions, while giving a barely perceptible flicker on a lamp, can play havoc with a calculation in a computing machine. For this reason some designers prefer to isolate their machines from the mains by feeding them through a rotating generator which has sufficient inertia to smooth out any short-term irregularities. The preferred arrangement is to provide an alternator driven by a motor connected to the mains and to make use of conventional rectifiers with voltage-stabilizing circuits to provide the various voltages required by the machine. During periods when the motor alternator is out of action for servicing or repair the machine can be run direct from the mains.

Bad connexions inside the machine are a more serious source of intermittent faults, and cannot, unfortunately, be entirely eliminated. Even if all the soldered joints are made with the greatest possible care, and carefully inspected shortly after being made, it will nevertheless be found that as time goes on some of them will become intermittent.

In a computer it is better that a joint should fail completely rather than become intermittent. The opposite is the case in most communication equipment, since it is better for a circuit to become noisy than to fail completely. It is probable, therefore, that standards for making and testing joints, different from those at present adopted in the electronics industry, will be developed for the manufacture of computers. It has been suggested that the adoption of new techniques, such as the use of printed or etched circuits, will eventually enable the incidence of intermittent faults to be reduced, but this has not yet been shown to be the case. Certainly it is necessary to accept the fact that, at the present time and in the foreseeable future, a substantial number of the faults occurring in a large electronic computer will be due to faulty connexions. Even if the machine is provided with built-in error-detecting circuits which indicate immediately when, and to some extent where, an intermittent fault has occurred, the location and correction of these faults are likely to waste a good deal of time. Their complete elimination or control remains an unsolved problem for the future.

Plugs and sockets are a further potential source of intermittent contacts, and widely differing experiences have been reported. Some

designers take the view that the use of plugs and sockets should be avoided altogether, even if this means that it takes some time to replace a faulty unit. Other designers, whose experiences with plugs and sockets have not been so discouraging, consider that the advantages of ready interchangeability of units are so great that the occurrence of occasional faulty contacts should be accepted. Naturally, much depends on the design of the plugs and sockets concerned, and the way in which they are used; even a good plug and socket can give trouble if it is subjected to mechanical strain. It must be remembered, when evaluating reports on the behaviour of plugs and sockets, that, when trouble occurs in a plug-in unit, there is always a tendency to put the blame on the plug-and-socket connector whether it deserves it or not. Bad contacts between valve-pins and valve-holders also account for a certain number of intermittent faults. There is undoubtedly much to be said for the use in computers of wired-in valves.

The majority of faults which occur in electronic computers fall into one of the categories which have been discussed. During any extended period, however, there will always occur an appreciable number of faults which do not fall into one of these categories. Many of these will be of a mechanical nature and such that they are extremely unlikely to recur. Individually, such faults can be ascribed to bad luck, but, collectively, they have an appreciable influence on the serviceability of the machine. All that can be done to lessen their rate of occurrence is to insist on an extremely high standard of workmanship in everything that goes into a computer, and to maintain this standard when alterations or repairs are made.

TRENDS IN PHYSICAL DESIGN

The constructional techniques used for electronic computers have generally been more or less closely modelled on those used in communication engineering. There has, however, been a tendency to mount more valves on each chassis, and this has led designers to give special attention to cooling. Otherwise such departures as there have been from conventional practice have mostly been introduced with a view to making it possible for maintenance work on the machine to be carried out rapidly, since, as I have already pointed out, computers differ from other large electronic installations in that, when one unit is out of action, the whole machine is generally useless. A recent trend, tending away from conventional methods of construction, is towards the creation of 'packaged' units, each containing several gates and perhaps one or more flip-flops, which can be used as logical elements in the construction of computers. The Bureau of Standards in the United States, and Elliott Brothers (London) Ltd., in England, have been pioneers in this development. Fig. 7·1 shows some units

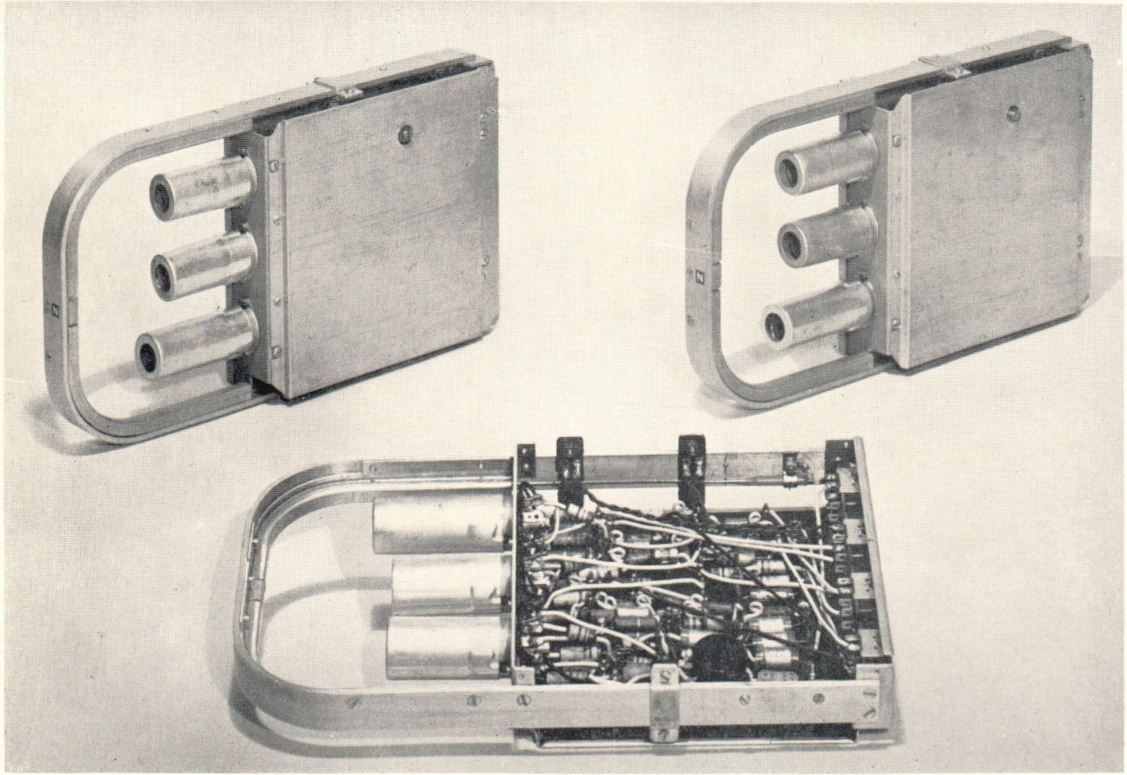


FIG. 7·1.—EARLY EXAMPLES OF PACKAGED UNITS FOR A SMALL COMPUTER (ELLIOTT BROTHERS (LONDON) LTD.)

used in a small magnetic drum computer built by the latter firm. Apart from the magnetic drum circuits, and one or two other special units, the whole computer was composed of packaged units of five different types, all being similar in mechanical construction to those shown in Fig. 7·1. The DYSEAC, a computer built by the Bureau of Standards using the same circuit techniques as the SEAC, makes use of small packaged units with printed circuits. The machines just mentioned are serial in operation; packaged units can also be used in parallel machines, but here there are perhaps advantages in using larger units containing 40 or 50 valves.

It is to be expected that the coming of transistors will make a revolution in computer design as in other branches of electronic engineering. The advantages of compactness and extremely low power consumption are obviously of great importance as far as digital computers are concerned. Already small experimental computers, composed entirely of transistors, are being built—for example, at Manchester University—but it will no doubt be some time before it is possible to design a machine using transistors which can equal in speed or reliability the more ambitious machines using vacuum tubes. The small size and low power consumption of transistors will, if a satisfactory degree of reliability can be obtained, make it possible ultimately to build machines containing far more logical elements than is at present practicable. It is in this respect primarily that the advent of transistors is likely to influence the development of logical design.

THE CHOICE OF AN ORDER CODE

The choice of an order code for a digital computer involves many considerations, some concerned with the engineering design of the machine and some with programming. A fundamental question to be settled is how many addresses the orders are to contain; the usual choice is between a single-address and a three-address order code, although special considerations associated with the techniques used in building a particular machine, or with some feature of its logical design, may make another type of order code attractive. On the whole, the straight choice between a single-address and a three-address order code is largely a matter of taste as far as the programmer is concerned.

In machines which use magnetic drums or ultrasonic delay units for the main store, an additional address specifying the storage location containing the order next to be executed is often provided, since this makes minimum-access coding possible. The use of minimum-access coding puts an extra load on the programmer, but the economic advantage obtained can be very great, since it may enable a machine fitted with a slow serial access store to compete, as regards speed, with one fitted with a rapid random access store. In the case of a

machine using a magnetic drum, even relatively inefficient minimum-access coding can sometimes enable a gain in speed of a factor of 20 or more to be obtained. It is likely, therefore, that minimum-access coding will be cultivated as long as a magnetic drum remains the cheapest form of store to use in the relatively inexpensive machine. As soon as some form of store which provides rapid random access is developed to a point at which it competes, on economic grounds, with the magnetic drum, there will be no further point in building machines designed for minimum-access coding.

The minimum number of distinct (single-address) operations which a machine must be able to perform in order to enable any mathematical calculation to be programmed is less than six. A machine with no more than the minimum number of orders, however, would be very inconvenient to programme—although the inconvenience might be largely mitigated by the use of subroutines and a carefully planned conversion routine—and it would also be very slow in operation. Convenience in use and speed in operation, therefore, combine to impel the designer to provide more orders than the minimum that are absolutely necessary; the problem is to decide how far this extension should be carried. Naturally, the more operations that are provided, the larger and more complex will the arithmetic and control sections of the machine become. This will have its effect on reliability and on the problems of maintenance.

The early universal high-speed machines to be completed naturally had fairly restricted order codes. These included orders for addition, subtraction, multiplication, the transfer of numbers to and from the store, input and output, a logical operation ('and', 'or', or 'not equivalent to'), conditional operations, and perhaps an order for facilitating the modification of addresses. Division was sometimes included and sometimes not. Experience showed that a machine with facilities on this scale and having between 16 and 32 orders in its order code could be an entirely practical instrument on which to base a computing organization, being neither too difficult to programme nor too slow in operation in relation to the amount of equipment provided.

In the last year or two the desire for higher speeds has tended to make designers look with favour on more comprehensive order codes, especially on those which include more elaborate facilities for modifying orders and for counting; these latter enable the amount of time the machine spends on operations connected with the organization of the arithmetic, instead of on the arithmetic itself, to be reduced. At the same time designers are tending to provide more elaborate arithmetic facilities, including built-in floating-binary or floating-decimal facilities. These had already been provided with great success in relay machines, and it was only the complexity of the control circuits required which prevented their being incorporated in

electronic machines, although this was not perhaps always openly admitted and recognized. Orders for automatic conversion from and to the scale of ten have already been provided in some binary machines, and are likely to become more popular. Possibly the provision of square-root extraction as a built-in operation will be revived—it was included in the ENIAC—but, apart from this, it does not seem likely that there will be any further major additions to the order code of a machine intended for general-purpose computing.

It is not, however, only the presence or absence of orders for performing major operations which affect the speed of a machine; every new order introduced into the code gives increased speed if it enables some operation to be performed with fewer orders than would otherwise be necessary. For example, some machines have orders which enable the product of two numbers taken from the store to be added into the accumulator; in these machines, formation of quantities of the form $ab + cd + ef$ can be effected with a minimum number of orders. Other machines have an order which enables the content of the accumulator to be multiplied by a number from the store and the product left in the accumulator; these machines lend themselves to the rapid computation of quantities like $abcd$. A machine provided with orders of both types would, on the average, be slightly faster than a machine provided with only one.

In the same way, it is in small matters rather than in big ones that the convenience of the programmer can most be served. It is not much more trouble for him to write down orders to call in a subroutine for (say) extracting a square root—especially if this and other commonly required subroutines are kept in some form of auxiliary store permanently accessible to the machine—than to write down the order or orders necessary to call into action a built-in unit. On the other hand, it does help the programmer—to take an example—if he is provided with a reasonable choice of conditional orders, each of which transfers control if a slightly different condition is satisfied. If he has only one conditional order he will find himself continually rewriting sections of programmes in which discriminations occur, in order to arrive at a discrimination of the only available form without using an excessive number of orders. Another way in which the designer of a machine can smooth the path of the programmer—especially a programmer without experience of the particular machine concerned—is by making the order code and its use as straightforward as possible. Any oddities among the orders, or illogical exceptions to the rules for using them, should be avoided, even if it means that the control circuits become slightly more complicated.

Extra orders which are in the nature of variants on existing orders can often be added without increasing appreciably the total amount of equipment in a machine. In any given design, however, a certain

number of digits only can be allocated for specifying the function of an order, and this will result, within the framework of that design, in a limit being set to the total number of orders which may be provided. The number of function digits available depends on the total number of digits in an order and on how many are needed to specify the address (or addresses), for B-registers, and for other purposes. In a machine in which orders and numbers are placed in the same store, the number of digits in an order must be equal to, or a simple submultiple of, the number of digits in a number. It will thus be seen that the number of orders in an order code is not a matter which can be considered only in relation to the programmer's needs, or to the complexity of the control circuits required to provide any particular set of facilities, but must also be considered in relation to the strategy and economy of the logical design of the machine as a whole.

THE USE OF CONVERSION ROUTINES

In Chapter 3 I described various programming techniques of some elaboration which enable a programme to be written in a form that is simple and straightforward from the point of view of the programmer, and afterwards to be converted automatically to the form required by the control circuits of the machine. The original programme can be written partly in symbolic form (using floating addresses) and without specifying where the various sections are to go in the store. Putting such a programme into the proper shape for the machine is a clerical task, carried out according to definite rules, and hence capable of mechanization. It is done by the machine, acting under the control of the conversion routine, before the programme is run for the first time; afterwards the programme may be stored in its converted form for future use. The utility or otherwise of elaborate conversion schemes is at present a matter of some controversy, and no general agreement seems likely to be reached for some time. The discussion given here must therefore be to some extent inconclusive.

A conversion scheme enables the programmer to be spared the necessity of slavish adherence to the particular form of order representation laid down by the designer. In particular, floating addresses enable alterations and corrections to be made in a programme without destroying its original orderly and logical layout, and without the risk of having to undertake wholesale renumbering of addresses and repositioning of subroutines. Without such a scheme it is a common experience that, frequently, the correction of one blunder introduces another. Again, when a conversion scheme is used, the code groups which are written down by the programmer can be arranged to have some mnemonic content, for example, by being recognizable abbreviations of the plain language descriptions of the functions for which they stand. Mistakes are less likely to be made in handling

such abbreviations than in handling arbitrary code groups in accordance with complicated, or seemingly illogical, rules. The use of a properly designed conversion scheme, say its advocates, enables programming to be learnt much more rapidly and surely by the newcomer.

There are, on the other hand, people who will have nothing to do with any form of conversion. They believe that the programmer does best to write his orders in a form as near as possible to that which they will take inside the machine—in the case of a binary machine this means the use of a scale of 8, 16, or 32—the input operation consisting of a simple transfer of digits to the store with, at the most, some assembly to form words. They maintain that to keep as close as possible to the form of orders inside the machine is the best way to avoid blunders, or at any rate to make it easy for them to be located. They admit that programming in this system may take longer to learn than in other systems, but maintain that this is of no importance, since the actual writing of the orders is only a small part of the programmer's task, the greater part being attention to mathematical planning and overall strategy. They say that it is better for a newcomer to make the effort of understanding thoroughly what goes on inside the machine, rather than to accept the shelter of an artificial system which can never be comprehensive enough to cover all contingencies, and which will not infrequently prevent the attaining of maximum efficiency. It is also pointed out by opponents of conversion schemes that the establishment of such a scheme absorbs a great deal of programming effort, and they suggest that this effort could better be directed towards solving problems. No sooner, they say, does someone get good at programming than he stops doing real programming and starts making programmes to make other programmes.

Let me say at once that for me it is not a question of whether or not to make use of some sort of conversion routine, but a question of how complicated the conversion routine should be and what facilities it should provide. After all, the conversion of numbers from decimal to binary form on their way into a binary machine is a form of conversion, and the initial orders in the EDSAC, which provide certain facilities, such as relative numbering of addresses in subroutines, form a simple conversion routine. Experience with these initial orders has left me in no doubt that programmers welcome, and make free use of, a system designed to provide them with extra facilities, if it is simple to use and if it forms part of the standard procedure for using the machine.

It is obviously very important to have a system of programming which is easily learnt, in the case of a machine which is available for use by a wide range of workers engaged on different problems. As

will be pointed out later, on p. 277, it is very desirable that research workers, making use of a machine as an aid to scientific investigation, should be in as close personal contact with the machine as possible, and this means that they must learn to do their own programming. Ease of training is less important in a computation centre with a purely professional staff of programmers whose main function is to provide a computing service for outside clients.

The great length and complexity of a comprehensive conversion routine is certainly a powerful drawback. Developing a conversion routine is to some extent like developing a computer, and can absorb a great many man-months of programming effort. Certainly it is true to say that up to date this effort could have been more productively employed in real programming, but it could similarly have been said a few years ago that, if all the people engaged in building computers had been set to compute with desk machines, more computing would have been done. As soon as conversion routines are out of the development stage, the amount of time taken to prepare one for a given computer will be of less importance. In the case of machines which are manufactured on a production basis the logical thing is for the manufacturers to provide the conversion routine; the same routine can then serve a large number of identical computers. However, this assumes that users will be content to use the conversion routine they are given. It might turn out that the temptation continually to modify the conversion routine, with the object of improving the facilities it provides, would be too strong to be resisted. If this proved to be the case, much programming effort would be frittered away, and the object of using a conversion routine mainly to make programming simple and fast would be defeated. There is something of a problem here, since programmers of the better type will always want to make the machine do as many of the routine tasks connected with programming as possible, and, if a stop is put to their activities in this direction, they may well decide to leave the field altogether. This is to some extent an answer to the objection that programmers, as soon as they become skilled, tend to put their efforts into making conversion routines rather than into real programming.

A conversion routine must have its own error-diagnosis routines associated with it, since a converted programme differs so much from that written by the programmer that conventional methods of error diagnosis are inclined to be awkward to apply. It is possible, for example, to construct a post-mortem routine which prints out the contents of the store in the same form (including the floating addresses) as that in which it was written. Such routines are, however, quite complicated, and the necessity to use them increases appreciably the overall complexity which is the great drawback of conversion schemes.

Even on a fast machine, the time taken to convert a fairly long programme may be several minutes; much use is made of the auxiliary store during conversion, so that the time greatly depends on the type of auxiliary storage provided. One of the main objects of using a conversion routine is to enable minor corrections to the programme to be made easily. This object is lost if conversion takes so long that there is a temptation to make corrections to the programme in its converted form rather than in its original form.

There is no reason why a programme which has been produced by a comprehensive conversion routine should, in ordinary cases, be slower in operation than one which has been written directly; in fact, the 'compiling routines' used in conjunction with the UNIVAC frequently give greater speed. This is obtained largely by using, instead of the usual repetitive loops, sequences of orders repeated (with the necessary modifications) a large number of times; as explained on p. 120, the system by which orders are transferred from the auxiliary to the main store in the UNIVAC makes this method of operation particularly favourable.

Less elaborate conversion routines may lead to a substantial loss of effective machine speed. However, this should not be regarded as a fatal objection to their use if they really do make programming simpler. The speed of most programmes is effectively determined by the speed at which a relatively short 'inner loop' is performed. If this inner loop is programmed directly and made as fast as possible, the use of the conversion routine for the remaining part of the programme will not lead to serious loss of speed. Even if this is not done, however, the use of the conversion routine may be justifiable in the case of a calculation which requires only a small consumption of machine time. In such cases the amount of time ordinarily spent in locating blunders in the programme may be substantially greater than that required for production. A conversion routine which enables the former to be cut down may, therefore, lead to a net saving in machine time. Even if the total amount of machine time consumed is greater than it otherwise would be, the fact that the cost of programming is reduced may make the use of a conversion routine economic. The validity of these arguments depends, however, on the extent to which programming can be facilitated by the use of conversion routines, and, since general agreement has not yet been reached on this point, it would be unwise to claim more than a contingent economic advantage for their use. It must be remembered also that machine designers are now giving increasing attention to ease of programming, particularly in relation to the choice of the order code.

When machines were first produced in more than isolated specimens, schemes of simplified programming, based on the use of interpretive routines, enjoyed a certain vogue. These caused the running

time of a problem to be increased as much as tenfold, and had the further disadvantage of occupying a good deal of storage space during the whole time the programme was being run. Since similar results can be obtained by the use of relatively simple conversion routines, which do not give the same loss in effective machine speeds or permanently take up space in the store, it seems likely that the use of interpretive methods for the purpose of simplifying programming will fall out of use.

THE ORGANIZATION OF A SCIENTIFIC COMPUTING CENTRE

Most of this book has been concerned with the design of digital computers and with the purely technical problems encountered in their maintenance. Once a satisfactory machine has been acquired, however, much remains to be done by way of organization before efficient use can be made of it. What form this organization should take, and how elaborate it should be, depends on the type of work to be undertaken, the type of machine or machines available, and whether the computing centre is an independent unit or whether it forms part of a larger research organization. My intention in writing this section is to do no more than indicate some of the considerations which should be borne in mind when setting up a computing centre. On the whole I have in mind a centre equipped with one of the larger machines.

The cost per hour of operating a large machine is quite high, and it is of the utmost importance that the available machine time should be used as efficiently as possible. In addition to its high intrinsic value, machine time often acquires a scarcity value as well. This was particularly so in the past, when the number of machines in operation was small. Even in the future, machine time will acquire a scarcity value in a given computing centre when it happens that there is more work to be done than can be handled conveniently on one machine, yet not enough work to justify the installation of a second machine.

In a centre where new work is continually being undertaken it is inevitable that a good deal of machine time will be used in checking programmes, and something has already been said about the necessity for insisting on the use of efficient methods for doing this. It is just as important that the programmes used for 'production' should be efficient. Often, more work on a programme—and this means more machine time spent on checking it—can lead to spectacular reductions in production time; factors of five to one are common. The ratio of time spent on production to that spent on checking programmes does not, therefore, give a good indication of the efficiency with which a machine is being used.

The ratio of production time to programme checking time depends very much on the type of problem being solved. For example, in the

computation of ballistic tables it is likely to be high, and in the case of a computation relating to atomic or nuclear physics it is likely to be low, since much experimenting with different methods of approach may be necessary before the final production run can be attempted. A different form of organization is desirable for these two types of application. In the first, the problem, fully specified, may be presented to the computing centre, which will, in due course, deliver the results; nothing more than a normal amount of liaison is necessary between the client and the staff of the centre. In the second case it may not be possible to specify the problem at all fully in advance. There may be a presumption that a certain kind of numerical approach will enable the problem to be solved, but, in order to make progress, it may be necessary to do a series of experimental calculations with the aid of the machine. In cases of this kind it is virtually essential that the theoretical physicist, or other research worker who is charged with the investigation, should work at the centre while the work is being done. He will almost certainly learn to handle his own programming and come to function as though he were actually a member of the staff of the centre.

The Progress of a Problem

It might be thought that the solution of a straightforward problem would fall into three clearly defined stages, namely planning, coding, and running. In practice, however, the first two stages run into each other, since attempts to code a problem are bound to reveal features not allowed for in the original plan, which must therefore be revised. In the same way, the coding of a problem cannot be completed without making a number of trial runs. Even when a satisfactory programme has been arrived at, and production has started, it is more than likely that alterations will be needed from time to time as the several parameters or initial values are varied over the ranges for which solutions are required. It sometimes becomes apparent, after a certain amount of running, that a different method of tackling the problem would enable a much faster programme to be written. The programmer then has the awkward problem of deciding whether to continue with the original programme or to set to work on the new one. His decision is likely to depend, among other things, on the amount of programming effort available.

Since programmers are so liable to make mistakes, it might be thought advantageous for them to work in pairs and check each other's programmes. Where this can be done it is undoubtedly the case that less machine time is spent in locating blunders than when programmers work independently. Unfortunately the highly detailed nature of the work of programming renders joint working impracticable in most circumstances, and programmers usually do better

and faster work on their own. It is frequently possible, however, for the programmer in charge of a problem to split off certain sections of the calculation and to draw up specifications for subroutines to perform them. These subroutines can then be constructed and tested by another programmer. In this way it is possible to bring the maximum of programming effort to bear on a particularly urgent problem. It is also possible to provide junior or inexperienced members of the programming group with tasks which are entirely within their capabilities, and at the same time sufficiently complete in themselves to be interesting.

It is important, if economy of machine time is the object, that the form of administrative control used should not be such as to make clients tend to overestimate their requirements. They will do this if they feel that, once their original demands have been met, they may have difficulty in regaining access to the machine to obtain supplementary results. A client who knows he can have use of the machine when he requires it will not, in general, ask for an extensive tabulation of a function, but will be content with the few values he requires for immediate use, together with a programme which will give him more when he wants them.

Staffing and Organization

There are three groups of people concerned with the running of a machine:

1. Maintenance staff.
2. Programmers.
3. Machine-room staff.

The maintenance staff may be either in attendance or on call, depending on the frequency of machine breakdown, which, in turn, depends on the size of the machine. Some machines are so designed that certain faults may be cleared by changing plug-in units or by switching over to stand-by equipment, and it is possible for members of the machine-room staff, given suitable training, to do this. It is necessary to set aside certain periods for 'scheduled maintenance'. During these periods the maintenance staff carry out systematically procedures—such as marginal checking—which have for their objects the forestalling of faults which would otherwise occur during periods of running. They also enable the maintenance staff to deal with minor troubles, mostly of a mechanical nature, which are not sufficiently serious to render the machine unserviceable but which reduce efficiency or which, if neglected, would lead to more serious trouble later on. A third and, at the present time, very important use which can be made of periods of scheduled maintenance is to make modifications which experience has shown are likely to improve reliability. For

example, it may have been found that a resistor used in a certain circuit is slightly under-rated; once this has been recognized, it is desirable to replace corresponding resistors in all similar circuits in the machine. Work of this kind may be described as 'constructive' maintenance, and it is important to allow the maintenance staff reasonable opportunities for doing it, not only because of its direct effect on machine reliability but also because the morale of the maintenance staff will suffer if they are not allowed to make small modifications to parts of the machine which they feel are unsatisfactory in design and could easily be improved. As the design of machines gradually becomes refined, however, the need for modifications of this kind will no doubt decrease.

It is very important that a close watch should be kept on maintenance procedures to make sure that everything which is done under the heading 'preventive maintenance' really has a material influence on machine serviceability. However clear it may appear on general grounds that a particular procedure is advantageous, its effect should, every now and then, be examined critically in the light of the serviceability record of the machine. The amount of time which can be spent on preventive maintenance is limited, and it may sometimes be advantageous to drop a procedure which has proved quite useful in practice in favour of another procedure which has proved on trial to be even more useful, even though the second procedure does not cover the same types of fault as the first. I am quite sure that a good deal of what is now done as preventive maintenance would not stand up to a critical evaluation based on results obtained.

Normally the machine should be in charge of a machine-room supervisor, who supervises the putting on of programmes. In some organizations the operation of the machine is left entirely to the programmers, who run their own programmes. The objection to this is that there is a danger of the machine being left idle while the programmer thinks about the results he has just obtained, instead of it being put to work on the next problem. Whatever the form of organization adopted, programmers should be encouraged to be present while their programmes are being run, since they are more likely than the machine-room staff to be able to detect when things are going wrong, but they should also be encouraged to provide enough information to enable the machine-room staff to run the programmes in their absence if necessary. This information should include instructions about applying post-mortem and error-diagnosis procedures if the programme obviously fails to run correctly for a reason other than a machine failure.

When the machine-room supervisor suspects that the machine has developed a fault he puts on one of the standard test programmes. If this confirms that there is a fault, and he cannot deal with it himself,

he should hand the machine over at once to the maintenance staff. There should be no compromise on this. It is often possible to get some results out of an unserviceable machine, but it is usually inefficient to do so; moreover, if the maintenance staff come to feel that some standard of serviceability other than the best that can be obtained is acceptable, their work is likely to suffer, and the machine serviceability will get worse and worse. Any form of organization, therefore, which encourages programmers to attempt to get results out of a partially unserviceable machine—for example, one which causes them to lose their turn on the machine as a result of handing it over—should be avoided.

Nothing will be said here about the number of people required in each of the three categories listed above, since everything depends on the type of centre and its equipment. In any case, the same considerations apply as to other staffing problems. One should err on the side of generosity in the provision of staff because of the high value which attaches to machine time; it is better that the staff should occasionally be under-employed than that machine time should be wasted when the staff are hard pressed.

For a similar reason a generous amount of auxiliary programming equipment should be provided, whether the input medium is punched tape, punched cards, or magnetic tape. This kind of equipment is relatively inexpensive—at any rate compared with the cost of a large digital computer—and very liable to mechanical breakdown. The standard of maintenance required in the case of teleprinters and similar items is somewhat higher than when these devices are used for their normal purposes.

THE USE OF DIGITAL COMPUTERS FOR BUSINESS PURPOSES

Something will now be said about the use of digital computers for performing the calculations and manipulation of data required in the day-to-day conduct of business. This includes the preparation of pay rolls, trading and stock analyses, the preparation of invoices and accounts, &c. It does not include calculations which, although associated with business, are indistinguishable from scientific calculations, for example, actuarial calculations performed in an insurance office.

In the past, calculating machines used in offices have been either desk machines, operated from keyboards, or punched-card machines working on fixed repetitive cycles and possessing very little internal storage capacity. The latter are now being supplemented by machines with a small number of storage registers and capable of performing simple programmes with some ten or twenty steps. These machines will undoubtedly be developed to have more storage capacity and more flexible programme control, and will become of great importance in office management. They will, however, for the most part re-

main essentially specialized in character, although some will approach the status of general-purpose machines, and thus bridge the gap which at present exists between business machines and large digital computers.

If a large digital computer is to be used economically in an office, it must justify its cost by performing a substantial fraction of the clerical work done in the office; that is, full use must be made of its flexibility as a general-purpose machine. The full-scale utilization of such computers for business purposes lies in the future, and all that can be done here is to make a few general remarks about what would appear to be the leading issues.

Business applications of digital computers involve the handling of large quantities of information with relatively little computation. This emphasis on data handling means that, in designing a computer for business applications, one should first decide what is the most satisfactory input and output system to use and then design the computer round it. However, although the amount of computation may be small compared with that encountered in scientific applications, it is not always contemptible; for example, the preparation of a pay roll necessitates the calculation of payment for work at hourly rates and piece rates, payments for overtime, bonuses and deductions for income tax, National Health or Social Security, contributory pensions, savings accounts, &c., together with the necessary totals, sub-totals, and cash dissections.

In conventional office organization the tendency is to divide up the work according to operations—for example, one clerk or group of clerks will do all overtime calculations, another clerk or group of clerks will calculate all tax deductions, and so on. A similar kind of organization is used in offices equipped with conventional punched-card machinery. When a large digital computer is used, however, the work is likely to be organized somewhat differently. For example, in the wage calculations just mentioned, all the steps leading to the computation of the sum due to a particular man would be carried out in a group, after which the machine would proceed to make the same calculation for the next man, and so on.

Although nearly all business calculations require a good deal of input and output, they do not require nearly as much as a casual glance at a room in which a large number of clerks are at work might suggest, since many of the figures written down by the clerks are intended only to be read by other clerks, and never go outside the office in which they are written. When the work is performed by an automatic machine instead of by clerks, intermediate results which will be needed later in the same machine run can be retained in the high-speed store or transferred to the auxiliary store. Results which will not be required again for some time can be put temporarily out of the machine

in a suitable form; magnetic tape or punched cards lend themselves well to this purpose. Printing is not normally necessary, and the tape or cards may simply be stored until the information is again required inside the machine. It is often found, however, that the subsequent programme can be greatly simplified, and the whole operation speeded up, if the material is re-ordered or sorted before being put back into the machine. This is because the use of a digital computer for sorting is relatively inefficient. In the same way it is frequently advantageous to sort, or arrange in a preferred order, new information before it is presented to the machine. The planning of such external sorting and ordering operations often plays a large part in the preparatory work which must be done before a large-scale digital computer can be used in a business application.

It has been pointed out earlier that the roles of input and output, temporary output and re-input, and auxiliary storage are closely related. This is particularly true of business applications, and it may well happen in a particular machine that the same devices are made use of for more than one purpose. For example, magnetic tape mechanisms may be used for all three, while card punches and card readers may be used for the first and second.

There is a great variety in the large routine clerical operations for which a digital computer might be used. Many of them have, however, one essential feature in common, namely, the maintenance of a file or ledger containing a large number of records. All the clerical operations performed are based in some way on the use of this file. In the case of stores record keeping, for example, the file might contain an entry for each individual item in stock showing: (*a*) the stock position at the end of each month, going back for, say, a year; (*b*) when new supplies were ordered; (*c*) when the goods were received; and (*d*) any other information which might be required in a particular trade. Similarly, a file used for accounting and invoicing might contain an entry for each customer, giving sufficient information to enable itemized accounts and invoices to be prepared.

If clerical operations of this class are to be fully mechanized the file must be held in some form of auxiliary store. The operations which have to be performed by the machine naturally depend on the particular circumstances, but, generally speaking, they will fall into the following subdivisions:

1. Keeping the file up to date by making postings in accordance with information received.
2. Initiating action according to the state of each entry at specified intervals of time.
3. Initiating action arising out of the posting of new information.

4. Following up earlier action.
5. Providing summaries and analyses at specified intervals.
6. Extracting and providing, at short notice, information about individual entries in the file.

In the case of invoicing, postings in the file would consist of information relating to goods ordered or dispatched and to remittances received. The action to be initiated would consist of the preparation of advice notes, invoices, and periodical statements of account. It would also be necessary for the machine to indicate when any particular account exceeded the credit limit set for it, and to initiate the sending of further demands if accounts were not paid within a specified time. In the latter case the machine could produce, ready for signature, a letter based on a standard *pro forma.*

It is clear that in a large business, such as might warrant the use of a digital computer, a very large amount of storage capacity would be required to contain the file. At the present time the only way of providing sufficient capacity would be by the use of magnetic tape or magnetic wire. Since random access to such a store can only be obtained at the cost of much time consumed in hunting, it is, in practice, necessary that the work should be organized so that both the extraction of information from the file and the posting of new information are done during periodical scans of the whole file. This means that incoming information must be classified and placed for temporary storage in a store with relatively rapid access; the programme which does this must provide a system of indexing so that the information can be incorporated in the file at the next opportunity. The normal arrangement would be for the scanning of the file to proceed in the intervals between the processing of new data and the taking of action under one of the headings given above. In a very large installation, however, it might be worth while to provide sufficient equipment to enable the processing of new data and the scanning of the file, with consequent action, to proceed in parallel.

A system such as that described here is likely to find application only in undertakings which are conducted on the largest scale. Scanning of the file, therefore, will take some hours or even several days. In the case of a particular insurance office, Wiseman estimates, for example, that as many as 500 reels of tape would be required to hold all the relevant information relating to a group of 700,000 policies. In a test on a UNIVAC it was found that policies could be dealt with at the rate of two a second, and it was thought that this could be speeded up to four a second; each complete scan of the file would then take about 50 hours. It would not be economic to provide sufficient tape mechanisms for all the reels containing the file to be accessible to the machine at the same time; instead, a limited number of

mechanisms would be provided, and the reels would be placed on them, as required, by an operator.

The requirement listed under 6 above, namely the supply at short notice of specific information, is an awkward one to meet without undue loss of time in hunting. However, in any commercial organization it is likely that the management will insist on the provision of such a feature, partly so that the needs of senior executive officials may be met, and partly so that inquiries made in person, or on the telephone, by members of the public, or other persons outside the organization, may be dealt with without delay. On receipt of an inquiry it would be necessary for the operator to place the reel of tape containing the required information on a tape mechanism reserved for this use; the normal action of the machine would then be halted, and it would be caused to extract the required information from the file and to print it in whatever form was required. It is probable that, provided steps were taken to ensure that the maximum number of inquiries of this kind which could arise during the course of a day was not large, they could be dealt with without disturbing seriously the routine scanning of the file. Extensive use of the facility could, however, seriously affect the efficiency with which the machine was being used.

The organization of the work would, of course, be greatly simplified if the file could be held in a high-speed random access store. Each item of incoming information could then be dealt with as it arrived, and there would be no difficulty about providing information required at short notice. It seems unlikely, however, that there will be developed, in the near future, a system of storage capable of providing, with rapid access, as much storage as can at present be obtained by the use of magnetic tape or wire; even if such a system were developed its cost would most likely be high on account of the inefficient use which would be made of the access system. In business applications of computers there is thus always likely to be a strong economic incentive to organize the work so that the amount of random access storage required is reduced to a minimum.

In considering whether a large-scale digital computer could be advantageously employed in any particular undertaking, one of the first steps is to make a survey of the clerical work which is currently being done. This is by no means as straightforward a task as might appear, since in any very large office organization there is much decentralization and delegation of responsibility. It is most unlikely that there will be any one person who possesses, at the same time, an overall knowledge of the organization and objectives of the clerical department and a detailed knowledge of the flow of work from clerk to clerk. It will, therefore, be necessary to interview many different people at different levels in the organization in order to piece together

a detailed flow chart. It is very likely that the survey will bring to light features which could be improved even apart from the installation of a large automatic machine.

Clerical programmes are apt to be long and complicated, partly because of the *ad hoc* nature of much clerical procedure. If efficient use is to be made of a large machine, it is very desirable that the procedures used should be rationalized as far as possible, particularly in relation to the many minor decisions which have to be made during a day's work and which, in the ordinary way, would not need to be formalized at all.

It might be thought that a clerical programme, once drawn up, would remain unchanged for long periods. This, however, is unlikely to be the case. In addition to the occasional large modifications made necessary by major changes in procedure, it will often be found necessary to make minor modifications and corrections in order to meet situations not fully envisaged when the programme was written. Also, as time goes on improvements to the programme may suggest themselves; some of these, if adopted, may lead both to a substantial speeding up of the work and to the release of machine time for other applications. For these reasons any organization which hopes to obtain the maximum advantage from the use of a large computer must secure the services of an efficient programming staff.

THE CONTROL OF INDUSTRIAL PROCESSES

There has been for some years an increasing tendency to make use of automatic devices to control the functioning of industrial plants, particularly in the chemical industry. The usual type of control device has for its object the maintaining of some variable, such as a temperature, at a constant value. Thus a single device controls a single servo-loop, although a number of related loops may be controlled in the whole process.

In this section I shall discuss some possible applications which large digital computers may have in the control of industrial processes. Although very little has so far been done in this field, there can be no doubt that important developments are likely to take place during the course of the next few years.

The object which one may hope to achieve by utilizing a large digital computer for the control of a plant or factory differs essentially from anything which could be achieved by the use of conventional control devices. The object is, in fact, to mechanize one of the functions of management, or, to put it more precisely, to provide the plant or factory manager with information which will enable him to achieve maximum efficiency of operation without relying, at the worst, on guess work, or, at the best, on the intuition which comes with long experience.

A digital computer used to assist the plant manager to achieve maximum efficiency need not—and normally would not—physically control the operation of the plant; it would provide information on the basis of which the operating controls of the plant would be set by hand in the conventional manner. This provides an answer to a question frequently asked in connexion with the 'automatic factory', namely, 'What happens when the machine breaks down?' The plant manager simply carries on as best he can until the machine is repaired and, if only a short gap in time has to be bridged, the temporary loss in efficiency is likely to be small.

The problems in control encountered in different industries differ greatly, and it is therefore difficult to make any general statements, particularly since the subject is at present relatively unexplored. All that can be done is to mention some of the possible lines of development. In order to fix ideas it will help to consider a chemical factory in which a variety of raw materials are used to make a variety of products. By adjusting the operating condition, the plant can be made to yield the various products in different proportions, and, similarly, the proportions in which the available raw materials are used can be controlled. The object of the plant manager is always to operate the plant in such a way that, having regard to the prevailing prices of raw materials and to the selling prices of the products, the maximum commercial advantage is obtained. The scope for the exercise of the plant manager's acumen is clearly greater if the plant is a large one, in which many different processes are performed and many different products are produced, than if a single process only is involved.

In order to be able to take informed decisions about the operation of the plant, the plant manager must be supplied with up-to-date information about its current performance. Basically this information is derived from readings, taken at regular intervals, of pressure gauges, thermometers, flow meters, and other instruments, but a considerable amount of processing may be required in order to put the information into useful form. Here a digital computer might well find application. Preferably the instrument readings should be transmitted automatically direct to the computer, via an analogue-to-digital converter; in this way the inefficiency which would result from manual transcriptions of instrument readings would be avoided.

By making available up-to-date information about plant output and efficiency, the use of a digital computer would enable the plant manager to take early action to correct any undesirable trend. There is a further and less obvious advantage to be obtained from the use of a digital computer. If information about plant efficiency and output is available only at long intervals and with much delay, it is difficult to observe the effects of experimentally introduced changes in plant-operating parameters, since these tend to be masked by the

general drift in operating conditions which must inevitably take place. The use of a digital computer, by providing accurate and frequent information, would remove this difficulty and would enable experiments to be made more accurately and more rapidly than is at present possible.

If an adequate theory of the mode of operation of a plant exists, there is no reason why a digital computer should not be used to predict the behaviour of the plant under given conditions, instead of being used merely to analyse its behaviour in the past. Suppose, for example, that the behaviour of a distillation column can be represented by a set of some 20 or 30 first-order, non-linear, differential equations. If the computer is programmed to solve these equations it becomes possible to investigate the running of the column under any specified conditions. If, as may be expected, this can be done in a very short time compared with the time which would be taken to perform the corresponding experiment on the plant itself, an approximation to the optimum operating conditions can be obtained by trial and error. It might even be possible to establish an iterative procedure which would enable the trial and error to be performed automatically.

In simple cases it would perhaps be possible to obtain, by means of a series of runs on the machine, a set of curves from which the optimum settings of the controls appropriate to any given circumstances could be obtained by inspection. Further access to the machine would not then be required. It is more likely, however, that too many variables would be involved for this to be possible and that it would be necessary to make machine runs as a matter of routine during the operation of the plant.

The difficulty of setting up a sufficiently accurate mathematical model of the plant must not be under-estimated. Even if sufficiently detailed fundamental knowledge of the processes concerned were available, the establishment of the model would be a substantial piece of research work calling for the highest ability and technical knowledge. More usually the fundamental knowledge would be entirely lacking, although there might be scope for making use of equations which expressed empirically or semi-empirically derived relations between the parameters. It must, however, be remembered that in the past there has been little incentive to investigate mathematically the operation of industrial plants, since any results obtained would have been of little value. Investigations of this kind will, no doubt, receive greater emphasis if it comes to appear that the results are likely to be of operational importance.

It is possible that the method known as *linear programming* will be found to have applications in plant operation. This method (it should be noted that the use of the word 'programming' has no connexion with its use for the setting up of a digital computer) has so far been

applied principally to problems in economics and logistics. The system under investigation is represented by means of a set of equations or inequalities in which the various parameters enter linearly. The problem posed is to find values of the parameters which maximize (or minimize) a given linear function of the parameters, and general methods, by which this problem may be solved, are available. If it could be made to yield sufficiently accurate results, linear programming would provide a short cut by which a detailed mathematical investigation of the operation of a plant could be avoided. Being a general method, it would lend itself readily to modification when changes were made in the plant, and it could be applied to situations of great complexity; for example, the interaction of a number of plants could be allowed for and account could even be taken, to some extent, of the effect which changes in plant operation might have on the external economy. From the mathematical point of view, the calculations to be made in linear programming resemble matrix manipulation, and they can, therefore, be performed on machines similar to those used for other scientific calculations, provided that sufficient storage capacity is available.

APPENDIX

MACHINERY AND INTELLIGENCE

In popular accounts automatic calculating machines are frequently described as 'thinking machines' or 'mechanical brains'. The use of these expressions, which imply that such machines in some way possess the power of reasoning, is symptomatic of the existence of a widespread, though unacknowledged, desire to believe that a machine can be something more than a machine. A similar attitude appears to have been prevalent in Babbage's time, for Lady Lovelace had occasion in 1843 to write: 'The Analytical Engine has no pretensions whatever to originate anything. It can do whatever we know how to order it to perform.'

An attempt to discuss, on a serious plane, whether or not machines can, or ever will, be able to think encounters at once the difficulty of giving a satisfactory definition of the meaning of the word 'think'. Some writers have insisted that the power to store information and to perform operations of a conditional nature are sufficient to constitute thinking, but this seems to me to be stretching the use of the word unduly. On the other hand, it can be argued that 'thinking' is a word which can legitimately be used only of a human being, or possibly of an animal, and that no action of a machine, however remarkable, should be called thinking.

Turing suggested that one might avoid this linguistic difficulty, and yet retain the essential point of the question 'Can machines think?', by asking instead whether a machine could, in certain circumstances, impersonate a human being. Naturally, it would not be necessary for the machine to resemble a human being in personal appearance, and Turing supposed that an examiner stationed in one room would communicate with a machine situated in another room by means of a teleprinter circuit. The examiner would put a series of questions or observations to the machine, which would transmit its own observations in reply. The question is whether a machine could, in these circumstances, deceive the examiner into thinking that he was communicating with a human being. Naturally, the wider the range of subjects which it was permitted to discuss, the more exacting would be the test. The following dialogue is based on one given by Turing as an example of what a machine which was successful in this test might be able to do:

Examiner: Do you know the sonnet which begins,
'Shall I compare thee to a summer's day?'?

Machine: Yes.
Examiner: Would not 'a spring day' do as well or better?
Machine: It wouldn't scan.
Examiner: How about 'a winter's day'; that would scan all right.
Machine: Yes, but nobody wants to be compared to a winter's day.
Examiner: Would you say Mr Pickwick reminded you of Christmas?
Machine: In a way.
Examiner: Yet Christmas is a winter's day, and I do not think Mr Pickwick would mind the comparison.
Machine: I don't think you're serious. By a winter's day one means a typical winter's day, rather than a special one like Christmas.

The questions are supposed to be entirely unpremeditated, so that there is no possibility of the answers being permanently wired into the machine. No existing machine, of course, can perform in anything like this way, and many people will be inclined to reject out of hand the idea that a machine ever could. If, at some time in the future, a machine were made to do it, however, the use of the description 'thinking machine' might not be altogether inappropriate.

Although the idea that machines can think exerts a fascination over many people, there are others who reject forcibly any such suggestion. When automatic digital computers were less familiar than they are now, such people would often attack computing engineers for using technical terms based on physiological analogies; the use of such terms was said—unjustly—to imply a claim that the machines possessed human attributes. An example of such a term is the word 'memory', which is frequently used to denote the part of a machine in which orders and numbers are held; the word 'store' has been used instead in this book, partly to avoid such criticism, although other terms, also based on physiological analogies, such as 'read', 'write', and 'discriminate', have been used freely. It is an interesting fact that in other subjects in which technical terms based on physiological analogies are used similar attacks have not been made. The reason is perhaps partly because the question 'Can machines think?' is closely related to the question 'Is the brain a machine?'. This latter question immediately invites a discussion of the problem of free-will, and other related problems, which are associated in most people's minds primarily with religious teaching and controversy. This, perhaps, accounts for some of the emotion which is aroused when the possibility of a machine thinking is discussed.

Quite apart, however, from its speculative and philosophical interest, the question of the extent to which human behaviour can be simulated by machines is of some practical importance, since the more machines can be made to replace human beings, the greater will be the share of the world's work which they can undertake. A machine

does not need to come anywhere near to passing Turing's test in order to be useful, provided that it can replace a human being in some restricted field. There are many machines—for example, automatic lathes or automatic washing machines—which are capable of doing this. However, such machines merely perform over and over again a fixed cycle of operations from which they are capable of making no departures, however trivial. Universal digital calculating machines, on the other hand, are capable of performing very highly complex sequences of operations, and possess, by virtue of being able to perform conditional operations, a certain power of discrimination. It is natural, therefore, that the development of such machines should have the effect of stimulating interest in the subject of this Appendix. Since a universal machine can, if it is provided with suitable 'input' and 'output' devices, be programmed to simulate the behaviour of any special-purpose machine, the question of what machines can do is now better discussed in terms of writing a programme for a universal machine than in terms of preparing a design for a special-purpose machine. The power of universal digital machines to simulate other machines extends to the simulation of analogue machines, provided it is admitted that, as far as interaction with the outside world is concerned, the variables may be quantized, there being, of course, no limit set to the fineness of quantization. In the case of a given analogue machine, the writing of the equivalent programme for a universal machine might be quite difficult, and the programme might be very slow in operation; however, as long as we are concerned only with the question of whether such and such a thing could or could not, in principle, be done by a machine these considerations are of little importance.

There is no difficulty in programming a universal machine to perform any set of operations when once the rules for performing them have been stated explicitly; this is true however complex the rules may be and however many alternative procedures of a conditional type are specified. Human behaviour differs from typical machine behaviour in that human beings have the capacity for learning, and this capacity is in evidence in whatever they undertake. It is not easy to say precisely what is meant by learning or what corresponds to it in a machine. For example, it is possible to put into the store of a digital computer a programme for playing the game of Nim, and in a sense it may be said that the machine has then been taught to play Nim. This analogy, however, is not a close one; it is as though a child were taught the multiplication table by means of a surgical operation on its brain. A slightly more subtle approach is to programme the machine to ask a number of questions, and then to proceed according to the answers received. But this is not much better, since the various answers which can be given must all be envisaged

when the programme is written, and they pass into the machine, as does the programme, through one of the input devices with which the machine is provided. To say that the taking in of the answers to the questions is learning, but that the taking in of the original programme is not learning, is to make an artificial distinction between one species of input and another. Many of the so-called learning programmes which have been demonstrated on digital computers are subject to this objection, although some of them are quite impressive in action, and within their very limited fields give a colourable imitation of true learning. In one example, a machine was given a series of stimuli, each stimulus being of type A or type B. Under control of the programme the machine predicted whether the next stimulus would be of type A or type B. If the stimuli formed a simple sequence repeated over and over again—for example AABAABAAB . . .—and provided the repetition period was not too great, the machine would, after a short time, 'learn' to predict correctly the type of the next stimulus. The construction of a programme for this purpose would not present any difficulty to anyone with a little experience of writing programmes for a digital computer; it can be done by enumerating and providing for all possible sequences of stimuli with periods less than, or equal to, the prescribed maximum period. Given such a programme, however, the machine would be defeated if confronted with a sequence of stimuli of longer periods, even if the sequence were a particularly simple one; it would similarly be defeated if another type of stimulus, type C, were introduced into the sequence. The programme could readily be modified to allow for these possibilities at the cost of making it longer, and indeed there would be no limit to the number of tricks the machine could be made to play if the programmer were prepared to make the programme sufficiently long and complicated. He could not, however, allow for every situation with which the machine might be confronted, and the programme would remain essentially a 'specialized' learning programme, in the writing of which the programmer had visualized, with more or less clarity, every relevant situation which could arise. Given one of these specialized programmes, in fact, the machine is able to learn up to a point, but once that point has been passed it is helpless. Here it differs from a human being, who is able to learn and to go on learning.

A programme which would enable a machine not only to learn, but also to go on learning, and to adapt itself to each new situation as it arose, I will call a 'generalized' learning programme. If such a programme could be constructed it would presumably consist of a relatively small initial nucleus of orders, and would adapt and extend itself as circumstances required, replacing its subroutines and switching sequences by others of a more general kind, and creating new subroutines to perform fresh functions. There does not appear to be

any reason for believing that the construction of such a programme is inherently impossible; on the other hand, it must be said emphatically that so far nothing of the kind has been constructed, nor have any practical suggestions been made as to the lines on which one might be constructed. Given such a programme, however, the operator would be able to teach the machine any trick that happened to occur to him, and he might, given time and patience, even set about 'educating' it in the sort of way that a child is educated. The lack of sufficient storage capacity or speed of operation might set practical limits as to what could be done in this direction, but theoretically there would be no reason why the machine should not be educated up to a point at which it could imitate human behaviour in a sufficiently wide range of situations for Turing's test to be applied. There is, however, no object in pushing the resemblance to human behaviour too far, and it might be that the best methods to use for 'educating' a machine would differ from those used for educating a child.

Although limitations of speed and storage capacity would undoubtedly soon make themselves felt, it is not these so much as shortage of ideas which is at present holding up progress. Whether anyone will see a way to make a generalized learning programme I do not know; but if anyone does, I am convinced that the resulting programme will not be a mere elaboration of the specialized learning programmes which have been constructed up to date, but will be based on some entirely new and original idea. The really challenging thing to my mind is that the technical means already exist whereby a programme could be made to extend itself by the creation of new sequences of orders; indeed, this is already done on a small scale in practical programmes, for example, in the construction and planting of link orders. The basic problem is one of programming the machine so that it can plan the construction of sequences of orders for performing new operations, that is, operations that were not envisaged by the original programmer. If this could be done, there would, I feel, be no difficulty about the construction and planting of the orders themselves.

Although I have written about machines passing Turing's test as though this is a thing which might actually happen, I do not wish to give the impression that I think it will happen soon, or even that present lines of research, pursued competently, or even brilliantly, over a considerable period, will lead to its happening at all. On the contrary, every line of approach which has been tried up to the present has proved abortive as far as this particular objective is concerned. Perhaps a genius will arise and turn the whole subject inside out. Until this happens I see a bright future for machines in executive capacities, but I do not see them behaving like human beings.

BIBLIOGRAPHY

1. AIKEN, H. A., and HOPPER, G. M., The Automatic Sequence Controlled Calculator. *Electrical Engineering*, v. 65, pp. 384, 449, 522 (1946).
   [An abridged version of reference 40.]
2. ALT, F. L., A Bell Telephone Laboratories' Computing Machine. *M.T.A.C.*, v. 3, pp. 1, 69 (1948).
   [A general description, written from the point of view of a user, of the Bell Computer Model V.]
3. ANDREWS, E. G., The Bell Computer Model VI. *Electrical Engineering*, v. 68, p. 751 (1949).
4. AUERBACH, A. A., ECKERT, J. P., SHAW, R. F., WEINER, J. R., and WILSON, L. D., The BINAC. *Proc. I.R.E.*, v. 40, p. 12 (1952).
   [A binary computer built by the Eckert-Mauchly Corporation, the designers of the UNIVAC.]
5. BABBAGE, C., Passages from the Life of a Philosopher. *Longmans, London* (1864).
   [Babbage's own account of some of the more interesting events of his life, with several chapters on calculating machines.]
6. BABBAGE, H. P., Babbage's Calculating Engines. *Spon & Co., London* (1889).
   [A complete collection, edited after Babbage's death by his son, of Babbage's writings on calculating machines, together with other documents bearing on his work; the relevant chapters from *Passages from the Life of a Philosopher* are included.]
7. BARNES, R. C. M., COOKE-YARBOROUGH, E. H., and THOMAS, D. G. A., An Electronic Digital Computer Using Cold Cathode Counting Tubes (Dekatrons) for Storage. *Electronic Engineering*, v. 23, pp. 286, 341 (1951).
8. BAXANDALL, D., Mathematics I—Calculating Machines and Instruments (catalogue of the collections in the Science Museum, South Kensington, with descriptive and historical notes and illustrations). *H.M. Stationery Office, London* (1926).
9. BIRD, R., Computing Machines: Input and Output. *Electronic Engineering*, v. 25, p. 407 (1953).
10. BLOCH, R. M., CAMPBELL, R. V. D., and ELLIS, M., The Logical Design of the Raytheon Computer. *M.T.A.C.*, v. 3, p. 286 (1948).
    [The RAYDAC.]
11. BLOCH, R. M., CAMPBELL, R. V. D., and ELLIS, M., General Design Considerations for the Raytheon Computer. *M.T.A.C.*, v. 3, p. 317 (1948).
12. BOOTH, A. D., The Physical Realization of an Electronic Digital Computer. *Electronic Engineering*, v. 22, p. 492 (1950).
13. BOOTH, A. D., A Signed Binary Multiplication Technique. *Quart. Journ. Mech. & Applied Math.*, v. 4, Pt. 2, p. 236 (1951).
14. BOOTH, A. D., and BOOTH, K. H. V., Automatic Digital Calculators. *Butterworth's Scientific Publications, London* (1953).
    [Covers design of computers and programming.]
15. BOWDEN, B. V. (editor), Faster than Thought. *Pitman, London* (1953).
    [A symposium on digital computing machines. Part I gives an introduction to the theory and history of the subject; Part II gives a brief description of British machines current at the time of writing; Part III is devoted to the application of automatic computers. The chapters in Parts II and III are contributed by writers associated with various British projects.]
16. BRADBURD, E. M., Magnetostrictive Delay Line. *Electrical Communication*, v. 28, p. 46 (1951).
17. BRAINERD, J. G., and SHARPLESS, T. K., The ENIAC. *Electrical Engineering*, v. 67, p. 163 (1948).
18. BROOKER, R. A., and WHEELER, D. J., Floating Operations on the EDSAC. *M.T.A.C.*, v. 7, p. 37 (1953).
    [Floating-point subroutines.]

19. BROWN, J. H., Programming for the MAGIC I System. *University of Michigan* (report) (1955).
[A description of a conversion routine developed under the direction of J. W. Carr.]
20. BUCK, D. A., and FRANK, W. I., Non-destructive Sensing of Magnetic Cores. *Communication and Electronics* (*Trans. A.I.E.E., Pt. I*), v. 72, p. 822 (January 1954).
[A method of sensing based on the use of a magnetic field applied at right angles to the writing field.]
21. BURKS, A. W., Electronic Computing Circuits of the ENIAC. *Proc. I.R.E.*, v. 35, p. 756 (1947).
22. BURKS, A. W., GOLDSTINE, H. H., and VON NEUMANN, J., Preliminary Discussion of the Logical Design of an Electronic Computing Instrument. *Institute for Advanced Study, Princeton* (report) (1946, 2nd edition 1947).
23. BUXTON, L. H. DUDLEY, Charles Babbage and his Difference Engine. *Transactions of the Newcomen Soc.*, v. 14, p. 43 (1934).
[An essay based partly on a MS. life of Babbage inherited by the author from his grandfather.]
24. DENMAN, H. H., KOPLEY, E. S., and PORTER, J. D., Comprehensive System Manual. *Digital Computer Laboratory, Massachusetts Institute of Technology* (report) (1953).
[A description of a comprehensive conversion routine developed under the direction of C. W. Adams.]
25. ECCLES, W. H., and JORDAN, F. W., A Trigger Relay Utilising Three-electrode Thermionic Vacuum Tubes. *Radio Review*, v. 1, p. 143 (1919).
[An historic paper.]
26. ECKERT, J. P. See reference 4.
27. ECKERT, J. P., MAUCHLY, J. W., GOLDSTINE, H. H., and BRAINERD, J. G., Description of the ENIAC and Comments on Electronic Digital Computing Machines. *Moore School of Electrical Engineering, University of Pennsylvania* (report) (November 1945).
28. E.R.A., High Speed Computing Devices. *McGraw Hill, New York* (1950).
[Mainly concerned with electronic circuits.]
29. ESTRIN, G., The Electronic Computer at the Institute for Advanced Study. *M.T.A.C.*, v. 7, p. 108 (1953).
[Logical design.]
30. FORRESTER, J. W., Digital Information Storage in Three Dimensions Using Magnetic Cores. *J. Appl. Phys.*, v. 22, p. 44 (1951).
31. FREEDMAN, A. L., Elimination of Waiting Time in Automatic Computers with Delay-type Stores. *Proc. Camb. Phil. Soc.*, v. 50, p. 426 (1954).
[An analysis of the advantages and disadvantages of minimum-access coding and other methods of eliminating waiting time in computers fitted with magnetic drums or delay stores.]
32. GILL, S., A Process for the Step-by-step Integration of Differential Equations in an Automatic Digital Computing Machine. *Proc. Camb. Phil. Soc.*, v. 47, p. 96 (1951).
[A modified Runge-Kutta method.]
33. GILL, S., The Diagnosis of Mistakes in Programmes on the EDSAC. *Proc. Roy. Soc.*, A, v. 206, p. 538 (1951).
34. GLUCK, S. E., The Electronic Discrete Variable Computer. *Electrical Engineering*, v. 72, p. 159 (1953).
[The EDVAC.]
35. GOLDSTINE, H. H., and GOLDSTINE, ADELE, The ENIAC. *M.T.A.C.*, v. 2, p. 97 (1946).
[A general description, including examples of how programmes are plugged.]
36. GOLDSTINE, H. H., and VON NEUMANN, J., Planning and Coding of Problems for an Electronic Computing Instrument. *Institute for Advanced Study, Princeton* (report), Part II, Vol. I (1947); Part II, Vol. II (1948).
37. HAMMING, R. W., Error-detecting and Error-correcting Codes. *Bell System Tech. J.*, v. 29, p. 147 (1950).

38. HARTREE, D. R., The ENIAC, an Electronic Computing Machine. *Nature*, v. 158, p. 500 (1946).
[A general description.]
39. HARTREE, D. R., Calculating Instruments and Machines. *University of Illinois Press* (1949); also *Cambridge University Press* (1950).
[Based on lectures given at the University of Illinois.]
40. HARVARD UNIVERSITY COMPUTATION LABORATORY. A Manual of Operation for the Automatic Sequence Controlled Calculator. *Annals of the Computation Laboratory, Harvard University*, v. 1 (1946).
[Includes a detailed description of the logical design and much engineering detail.]
41. HARVARD UNIVERSITY COMPUTATION LABORATORY. Description of a Relay Calculator. *Annals of the Computation Laboratory, Harvard University*, v. 14 (1949).
[A detailed description of the Harvard Computer Mark II.]
42. HARVARD UNIVERSITY COMPUTATION LABORATORY. Description of a Magnetic Drum Calculator. *Annals of the Computation Laboratory, Harvard University*, v. 25 (1952).
[A detailed description of the Harvard computer Mark III.]
43. HARVARD UNIVERSITY COMPUTATION LABORATORY. Synthesis of Electronic Computing and Control Circuits. *Annals of the Computation Laboratory, Harvard University*, v. 27 (1951).
44. HOPPER, G. M., The Education of a Computer. *Proceedings of the Association for Computing Machinery, Pittsburgh* (1952), p. 243.
[The use of compiling routines.]
45. HUSKEY, H. D., Characteristics of the Institute for Numerical Analysis Computer. *M.T.A.C.*, v. 4, p. 103 (1950).
[The SWAC.]
46. I.B.M., Type 650; Operating Principles. *International Business Machines Corporation, New York* (1953).
[A machine based on the use of a magnetic drum for the main store.]
47. I.B.M., I.B.M. Electronic Data Processing Machines. Principles of operation: Type 701 and associated equipment. *International Business Machines Corporation, New York* (1953).
48. I.B.M., Electronic Data Processing Machines: Type 702. *International Business Machines Corporation, New York* (1954).
[An alpha-numeric (decimal) computer with Williams tube storage designed for business applications.]
49. I.B.M., Electronic Data Processing Machines: Type 704. *International Business Machines Corporation, New York* (1954).
[A later development of the I.B.M. Type 701 computer.]
50. KAYE, E. J., and GIBBS, G. R., LEO (Lyons Electronic Office): Part 3, a checking device for punched data tapes. *Electronic Engineering*, v. 26, p. 386 (1954).
[See references 76 and 57 for Parts 1 and 2.]
51. KEISTER, W., RITCHIE, A. E., and WASHBURN, S. H., The Design of Switching Circuits. *Van Nostrand, New York* (1951).
[Concerned mainly with relay switching circuits.]
52. KILBURN, T., The University of Manchester Universal High Speed Digital Computing Machine. *Nature*, v. 164, p. 684 (1949).
[Brief description of an experimental machine.]
53. KILBURN, T., TOOTILL, G. C., EDWARDS, D. B. G., and POLLARD, B. W., Digital Computers at Manchester University. *Proc. I.E.E.*, v. 100, Part II, p. 487 (1953).
[Detailed description, with circuits.]
54. KJELLBERG, G., and NEOVIUS, G., The BARK, a Swedish General Purpose Relay Computer. *M.T.A.C.*, v. 5, p. 29 (1951).
55. KNIGHT, L., Valve Reliability in Digital Calculating Machines. *Electronic Engineering*, v. 26, p. 9 (1954).
56. LEINER, A. L., and ALEXANDER, S. N., System Organisation of the DYSEAC. *Trans. I.R.E.*, v. EC–3, No. 1, p. 1 (1954).
[Logical design of a machine developed from the SEAC.]

57. LENAERTS, E. H., LEO (Lyons Electronic Office): Part 2, operation and maintenance. *Electronic Engineering*, v. 26, p. 335 (1954).
[See references 76 and 50 for Parts 1 and 3.]
58. LOVELACE, COUNTESS OF. See reference 61.
59. LUDGATE, P. E., On a Proposed Analytical Machine. *Sci. Proc. R. Dublin Soc.*, v. 12, p. 77 (1909).
60. MCCULLOCH, W. S., and PITTS, W., A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bull. Math. Biophysics*, v. 5, p. 115 (1943).
61. MENABREA, L. F., Bibliothèque Universelle de Genève, No. 82 (October 1842).
[A translation, with extensive notes, by the Countess of Lovelace was published in Taylor's *Scientific Memoirs*, v. 3, p. 666 (1843). This translation was reprinted by H. P. Babbage (reference 6), and has more recently been reprinted by Bowden (reference 15).]
62. MICHAEL, F. R., Tube Failures in the ENIAC. *Electronics*, v. 20, p. 116 (1947).
63. MILLERSHIP, R., ROBBINS, R. C., and DEBARR, A. E., Magnetostriction Storage Systems for a High Speed Digital Computer. *Brit. J. App. Phys.*, v. 2, p. 304 (1951).
[Short note describing principle of operation.]
64. MINNICK, R. C., Magnetic Switching Circuits. *J. App. Phys.*, v. 25, p. 479 (1954).
[Discusses the use of magnetic cores having rectangular hysteresis loops.]
65. MOORE SCHOOL OF ELECTRICAL ENGINEERING. Theory and Techniques for Design of Electronic Digital Computers. Lectures given to a special course at the Moore School 8 July–31 August 1946. *Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia* (report in 4 volumes) (1947–48).
66. MOORE SCHOOL OF ELECTRICAL ENGINEERING. The EDVAC—a Preliminary Report on Logic and Design. *Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia* (report) (1948).
67. MOORE SCHOOL OF ELECTRICAL ENGINEERING. A Functional Description of the EDVAC. *Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia* (report in 2 volumes) (1949).
68. MULLER, D. E., Application of Boolean Algebra to Switching Circuit Design and to Error Detection. *Trans. I.R.E.*, v. EC-3, No. 3, p. 6 (1954).
69. NATIONAL BUREAU OF STANDARDS. The Operating Characteristics of the SEAC. *M.T.A.C.*, v. 4, p. 229 (1950).
70. NEWMAN, E. A., CLAYDEN, D. O., and WRIGHT, M. A., ACE Store. *Proc. I.E.E.*, Pt. II, v. 100, p. 445 (1953).
71. OETTINGER, A. G., Programming a Digital Computer to Learn. *Phil. Mag.*, 7, v. 43, p. 1243 (1952).
72. OFFICE OF NAVAL RESEARCH. A Survey of Automatic Digital Computers. *Office of Naval Research, Washington, D.C.* (1953).
[Includes a brief specification of all computers known to be operating or under construction at the time of compilation.]
73. PAGE, C. H., Digital Computer Switching Circuits. *Electronics*, v. 21, p. 110 (1948).
74. PAPIAN, W. N., A Coincident-current Magnetic Memory Cell for the Storage of Digital Information. *Proc. I.R.E.*, v. 40, p. 475 (1952).
75. PAPOULIS, A., The Non-destructive Readout of Magnetic Cores. *Proc. I.R.E.*, v. 42, p. 1283 (1954).
[Describes further theoretical and experimental work on the method of reference 20.]
76. PINKERTON, J. M. M., and KAYE, E. J., LEO (Lyons Electronic Office): Part 1, history and technical description. *Electronic Engineering*, v. 26, p. 284 (1954).
[See references 57 and 50 for Parts 2 and 3.]
77. POLLARD, B. W., and LONSDALE, K., The Construction and Operation of the Manchester University Computer. *Proc. I.E.E.*, v. 100, Part II, p. 501 (1953).
[Includes much engineering detail.]
78. RAJCHMAN, J. A., The Selectron—a Tube for Selective Electrostatic Storage. *M.T.A.C.*, v. 2, p. 359 (1947).

79. ROBINSON, A. A., Multiplication in the Manchester University High Speed Digital Computer. *Electronic Engineering*, v. 25, p. 6 (1953).
80. SHANNON, C. E., A Symbolic Analysis of Relay and Switching Circuits. *Trans. Amer. Inst. Elect. Eng.*, v. 57, p. 713 (1938).
81. SHANNON, C. E., Programming a Computer for Playing Chess. *Phil. Mag.*, v. 41, p. 256 (1950).
82. SHAW, R. F., Arithmetic Operations in a Binary Computer. *Rev. Sci. Inst.*, v. 21, p. 687 (1950).
83. STAEHLER, R. E., An Application of Boolean Algebra to Switching Circuit Design. *Bell System Tech. Journ.* v. 31, p. 280 (1952).
84. TAYLOR, N. H., Marginal Checking as an Aid to Computer Reliability. *Proc. I.R.E.*, v. 38, p. 1418 (1950).
85. TURING, A. M., Computing Machines and Intelligence. *Mind*, v. 59, N.S., p. 433 (1950).
[A penetrating and witty essay addressed to philosophers.]
86. VON NEUMANN, J., First draft of a report on the EDVAC. *Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia*, (report) (June 1945).
87. WANG, AN, and WOO, WAY DONG, Static Magnetic Storage and Delay Line. *J. Appl. Phys.*, v. 21, p. 49 (1950).
88. WEST, C. F., and DETURK, J. E., A Digital Computer for Scientific Applications. *Proc. I.R.E.*, v. 36, p. 1452 (1948).
[The RAYDAC.]
89. WHEELER, D. J., Programme Organisation and Initial Orders for the EDSAC. *Proc. Roy. Soc.*, A, v. 202, p. 573 (1950).
90. WIDROW, B., A Radio-frequency Nondestructive Readout for Magnetic-Core Memories. *Trans. I.R.E.*, v. EC-3, No. 4, p. 12 (1954).
91. WIER, J. M., A High-Speed Permanent Storage Device. *Trans. I.R.E.*, v. EC-4, p. 16 (1955).
[A system based on the use of magnetic cores.]
92. WILKES, M. V., The ENIAC. *Electronic Engineering*, v. 19, p. 104 (1947).
[A short account of the logical design.]
93. WILKES, M. V., The Use of a 'Floating Address' System for Orders in an Automatic Digital Computer. *Proc. Camb. Phil. Soc.*, v. 49, p. 84 (1953).
94. WILKES, M. V., PHISTER, M., and BARTON, S. A., Experience with Marginal Checking and Automatic Routining of the EDSAC. Convention Record of the I.R.E., 1953, Pt. 7, p. 66; also Automatic Digital Computation (proceedings of a symposium held at the National Physical Laboratory in March 1953), p. 239, *H.M.S.O., London* (1954).
95. WILKES, M. V., AND RENWICK, W., An Ultrasonic Memory Unit for the EDSAC. *Electronic Engineering*, v. 20, p. 208 (1948).
96. WILKES, M. V., and RENWICK, W., The EDSAC—an Electronic Calculating Machine. *J. Sci. Inst.*, v. 26, p. 385 (1949).
[Logical design.]
97. WILKES, M. V., and RENWICK, W., The EDSAC (Electronic Delay Storage Automatic Calculator). *M.T.A.C.*, v. 4, p. 61 (1950).
[Logical design.]
98. WILKES, M. V., WHEELER, D. J., and GILL, S., The Preparation of Programs for an Electronic Digital Computer with Special Reference to the EDSAC and the Use of a Library of Subroutines. *Addison-Wesley Press, Cambridge, Mass.* (1951).
99. WILKINSON, J. H., An Assessment of Optimum Coding on the Pilot ACE. *Phil. Trans.* (in press).
100. WILLIAMS, F. C., and KILBURN, T., Electronic Digital Computers. *Nature*, v. 162, p. 487 (1948).
[A brief account of the first very simple experimental machine built at Manchester University.]
101. WILLIAMS, F. C., and KILBURN, T., A Storage System for Use with Binary-digital Computing Machines. *Proc. I.E.E.*, v. 96, Part II, p. 183 (1949); also *Proc. I.E.E.*, v. 96, Part III, p. 81 (1949).
[Cathode-ray tube storage.]

102. WILLIAMS, F. C., KILBURN, T., and TOOTILL, G. C., Universal High Speed Digital Computers; a Small-scale Experimental Machine. *Proc. I.E.E.*, v. 98, Part II, p. 13 (1951).
[A more detailed account of the machine described in reference 100.]
103. WILLIAMS, F. C., and WEST, J. C., The Position Synchronization of a Rotating Drum. *Proc. I.E.E.*, v. 98, Part II, p. 29 (1951).
104. WILLIAMS, F. C., KILBURN, T., and THOMAS, G. E., Universal High Speed Digital Computers: a Magnetic Store. *Proc. I.E.E.*, v. 99, Part II, p. 94 (1952).
[Detailed description, with circuits, of a store using a magnetic drum.]
105. WILLIAMS, F. C., ROBINSON, A. A., and KILBURN, T., Universal High Speed Digital Computers: Serial Computing Circuits. *Proc. I.E.E.*, v. 99, Part II, p. 107 (1952).
106. WILLIAMS, F. C., KILBURN, T., LITTING, C. N. W., EDWARDS, D. B. G., and HOFFMAN, G. R., Recent Advances in Cathode Ray Tube Storage. *Proc. I.E.E.*, v. 100, Part II, p. 523 (1953).
107. WISEMAN, R. T., Life Insurance Premium Billing and Combined Operations by Electronic Equipment. *J. Assn. Computing Machinery*, v. 1, p. 7 (1954).
108. WOODGER, M., Automatic Computing Engine of the N.P.L. *Nature*, v. 167, p. 270 (1951).
[A brief description of the pilot model of the ACE.]

Much information about digital computers is available in reports of conferences; of these the following are the most important:

PLACE	YEAR	TITLE OF REPORT	PUBLISHER	DATE
Harvard	1947	Annals of the Computation Laboratory, v. 16.	Harvard University Press	1948
Cambridge	1949	Report of conference on high speed automatic calculating machines.	University Mathematical Laboratory, Cambridge (*out of print*).	1950
Harvard	1949	Annals of the Computation Laboratory, v. 26.	Harvard University Press.	1951
Manchester	1951	Manchester University computer inaugural conference.	Computing Machine Laboratory, Manchester University.	1952
Philadelphia	1951	Review of electronic digital computers. Joint AIEE-IRE computer conference.	American Institute of Electrical Engineers, New York.	1952
Pittsburgh	1952	Proceedings of the Association for Computing Machinery.	Association for Computing Machinery, New York.	1952
Toronto	1952	Proceedings of the Association for Computing Machinery.	Association for Computing Machinery, New York.	1952
Los Angeles	1953	Proceedings of the Western Computer Conference.	Institute of Radio Engineers, New York.	1953
N.P.L., Teddington.	1953	Automatic Digital Computation.	H.M.S.O., London.	1954
New York	1953	Institute of Radio Engineers National Convention, Part 7.	Institute of Radio Engineers, New York.	1953
Los Angeles	1954	Trends in computers: automatic control and data processing.	American Institute of Electrical Engineers, New York.	1954
Washington	1954	Automatic programming for digital computers.	Office of Naval Research, Washington, D.C.	1954

PLACE	YEAR	TITLE OF REPORT	PUBLISHER	DATE
New York	1954	Institute of Radio Engineers National Convention, Part 4.	Institute of Radio Engineers, New York.	1954

Attention may also be drawn to *Proc. I.R.E.*, v. 41, No. 10 (October 1953), which was a special issue containing over 30 papers on digital computers and allied subjects.

The following current periodicals are of special interest to those concerned with computing machines. The first two contain abstracts of current literature.

Mathematical Tables and other Aids to Computation (*M.T.A.C.* or *Math. Tab.*; published by National Academy of Science—National Research Council, Washington, D.C.).

Transactions of the I.R.E. Professional Group on Electronic Computers.

Journal of the Association for Computing Machinery.

Computers and Automation (published by Berkeley Enterprises, New York).

The books listed under references 14, 15, 28, 39, 43, 51 and 98 cover, wholly or partially, the same field as this book. More recently published books on allied topics are:

RICHARDS, R. K., Arithmetic Operations in Digital Computers. *Van Nostrand, New York* (1955).

LOCKE, W. N., and BOOTH, A. D., Machine Translation of Languages. *John Wiley and Sons, New York*, and *Chapman and Hall, London* (1955).

INDEX

Figures in roman type refer to pages; those in italics refer to the Bibliography

ACE (pilot model), 76, 78, 152–3, 181; *70*, *99*, *108*
Adams, C. W., *24*
Adders and Accumulators
  binary, 34–6, 62–3, 212, 223–6, 232–3, 235, 237, (relay) 145–8, (Kirchhoff), 235–7,
  decimal, 242–50, (relay) 74
Address, 36,
  floating, 127, 272
Alexander, S. N., *56*
Alt, F. L., 137; *2*
Analogue computer, 1
Analytical Engine, 6–15
'And' circuit, *see* Gate
Aiken, H. H., 16, 192
Assembly of programme, 97–8, 126
Asynchronous operation, 67
Automatic Sequence Controlled Calculator, 16–20, 41, 68; *1*, *40*
Auxiliary storage, 36, 76–7, 95–6, 110, 119–21

Babbage, Charles, 2–15, 86, 289; *5*, *6*, *23*
Babbage, H. P., 5, 10, 13; *6*
BARK, *54*
Bell Telephone Laboratories, *51*
  computer development at, 133–7,
  Bell Computer Model V, 137–9; *2*,
  Bell Computer Model VI, 139–40; *3*
BINAC, *4*
Binary codes for decimal digits, 75, 239–42
Binary computing circuits, 34–6, 53–6, 61–5, 223–9; *73*, *105*
Binary-decimal conversion, 57–8
Binary scale, 31,
  advantage of, 33
Bistable circuit,
  flip-flop, 21, 22; *25*,
  counter, 230–3
Bi-quinary code, 135
Blunders, detection of, in programmes, 121–3, 277; *33*
Boolean algebra, 213–15
Booth, A. D., 177, *14*
B-register, 72–3, 85
Buffer circuit, *see* 'or' circuit
Bush, V., 1
Business applications of digital computers, 280–5; *107*

Cambridge University, *see* Mathematical Laboratory *and* EDSAC
Carr, J. W., *19*
Carry
  anticipatory, 13–15, 17, 23–4, 233,
  end-around, 41,
  successive, 14–15, 232
Cathode-ray display, 81
Checking of numerical results, 81–4, 124
Checking circuits, *see* Error detection
Clock pulses, 32
Coding circuit, 227–29
Coincident current matrix store, 194–201; *30*, *74*
Compiling routines, 275; *44*
Complement, 40–2, 54, 227
Complementing circuit, 54
Computation Laboratory, Harvard University, 5, 16–20, 71, 133–4, 192, 210, 215–16, 240, 246; *43*. *See also* Harvard machines
Conditional order, 11–13
Control combination, 61, 92
Control, transfer of, 42

Conversion routines, *19*, *24*, *44*,
description of, 124–31,
evaluation of, 272–6
Counters,
binary, 230–2,
decimal, 23, 239
Cyclic codes, 255–6

Decimal computers, 73–6,
choice of binary code for, 239–42
Decimal computing circuits, 23, 239–51
Decoder, 50, 202–4, 236
Dekatrons, used in computer, *7*
Delay circuit, 34
Desk calculating machines, 1,
development of, 4; *8*
Destination, 49
Difference Engine, 3–6
Differential equations, subroutines for solving, 113–14; *32*
Diode matrices, 222–3, 225, 248–9
Diode switching circuits, 219–23
Directory, 104–5
Dividers, binary, 237–9
Division, subroutine for, 44, 110–12
DYSEAC, 269; *56*

Eckert, J. Presper 20; *4*, *27*
EDSAC, 39, 42–61, 76, 84, 87–110, 153–7; *95*, *96*, *97*, *98*
EDVAC, 30–8, 153–4, 168; *34*, *66*, *67*, *86*
Electrostatic storage, 160–6, 166–8; *78*, *101*, *106*
Elementary functions, subroutines for, 112
Elliott Bros. (London) Ltd., 268
Engineering of computers,
viewed as branch of electronic engineering, 257–9,
trends in physical design, 268–9,
operating margin, 259–60
ENIAC, 20–30, 76, 236, 241, 251, 271; *17*, *21*, *27*, *35*, *38*, *62*, *92*
Error detection, circuits for, 81–4, 134–7, 148–50, 261–3
Error detecting and correcting codes, 58, 252–4; *37*
Error diagnosis, *see* Blunders
Excess-three code, 135, 240–1, 245–6

Farr, W., 5
Faults, 263–5, 266–8; *55*, *62*
Ferranti Ltd., 77, 164, 237
Ferrite cores, *see* Magnetic cores
Flip-flop, 21–2; *25*
Floating address, 127, 272; *93*
Floating binary arithmetic, 106, 114, 270–1; *18*
Floating decimal arithmetic, 106, 114, 137, 141, 270–1; *18*
Flow diagrams, 97–8
Forrester, J. W., 194; *30*

Gate, 21, 34, 189, 220–1, 234
Gill, S., *32*, *33*, *98*
Goldstine, H. H., 96; *27*, *35*, *36*
Gray code, *see* Reflected code

Half-adder, 225–6
Hamming, R. W., 253; *37*
Hartree, D. R., *38*, *39*
Harvard machines
Mark I, *see* Automatic Sequence Controlled Calculator,
Mark II, 134, 141–5; *41*,
Mark III, 71, 76, 241; *42*,
Mark IV, 71, 76, 80, 194
Harvard switching algebra, 215–16
Harvard University, *see* Computation Laboratory
Harwell, 239; *7*
Hopper, G. M., *1*, *44*
Huskey, H. D., *45*

I.B.M., 15
I.B.M. machines, *46*, *47*, *48*, *49*
I.B.M. type 701 data processing machine, 78–9, 80; *47*, *49*
Industrial process control, 285–8
Inhibiting circuit, 34
Initial input routine, 58–61, 93–6; *89*, *98*
Initial orders, 59; *89*, *98*
Input and output
sub-routines for, 56–58, 110–11,
different media compared, 76–81; *9*

Input transformation, 240
Institute for Advanced Study, Princeton, machine designed at, 64, 70, 84, 238; *22*, *29*, *36*
Instruction, *see* Order
Intelligence, 2, 289–293; *71*, *81*, *85*
Interlude, 108
International Business Machines Corporation, *see* I.B.M.
Inverting circuit, 34, 212

Jacquard loom, 7–8

Kilburn, T., 160–2, 164, 174–5, 181; *52*, *53*, *100*, *101*, *102*, *104*, *105*, *106*
Kirchhoff switching circuits, 233–6, 237

Lardner, D., 5
Leiner, A. L., *56*
LEO, 153; *50*, *57*, *76*
Library of subroutines, 86, 106–14
Linear algebra, subroutines for, 114
Linear programming, 287–8
Link order, 98–9
Logical design
  definition of, 20–1
  symbols used in discussing, 21–2, 33–4
Lovelace, Ada Augusta, Countess of, 10, 12, 289; *61*
Ludgate, P. E., 15; *59*

Magnetic cores
  used for switching, 189, 202–4; *64*,
  used for storage, 189, 192–201, 205–6,
  manufacture and testing of, 206–9
Magnetic drum, 176–85; *103*, *104*
Magnetic hysteresis, 190–2
Magnetic storage, methods of recording in moving media,
  principle of, 168–70,
  dipole method, 171–3,
  polarised dipole method, 175–6,
  non-return to zero method, 173–4,
  phase-modulation method, 174–5
Magnetic storage, static, 188–209; *20*, *30*, *74*, *75*, *90*, *91*
Magnetic tape, 79–80, 185–8
Magnetic wire, 188
Magneto-strictive delay unit, 157–60; *16*, *63*
Major cycle, 47
Maintenance, 278–80,
  preventative, 279
Manchester University, 72,
  digital computer at, 73, 76, 120, 237, 269; *52*, *53*, *77*, *100*, *102*
Margin of operation, 259–60
Marginal checking, 265–6; *84*, *94*
Massachusetts Institute of Technology, 81, 101, 131, 166–7, 194–8, 265; *see also* Whirlwind I
Master routine, 98–101
Mathematical Laboratory, Cambridge, 39; *see also* EDSAC
Mathematical tables
  subtabulation of, 3, 5,
  printing of, 5–6,
  use by digital computer of, 9, 112
Mauchley, John W., 20; *27*
McCulloch, W. S., 33; *60*
Menebrea, L. F., 10–12; *61*
Mercury tanks, 152–7
Minimum-access coding, 115–19; *31*, *99*
Minor cycle, 47
Moore School of Electrical Engineering, 20, 31, 160; *65*; *see* also ENIAC, EDVAC
Multipliers,
  binary, 64–65, 236–7; *13*, *79*
  decimal, 250–1

National Bureau of Standards, 188, 268–9; *see also* SEAC
National Physical Laboratory, *see* ACE (pilot model)
Negative numbers, representation of, 39–42
Newman, E. A., *70*

Optimum coding, 115–19; *31*, *99*
'or' circuit, 21–2, 34, 211, 220–2

Order, 2,
represented as number, 30–1, 44,
interpretation of, 69–70,
with 'algebraic' address, 71,
pseudo-, 89
Order code
single address, 42,
multi address, 36, 114–16

Papian, W. N., *74*
Parallel mode, 61,
compared with serial, 66
Parameters
programme, 102,
preset, 107
Parity, 252
Photographic recording, 81
Picture storage tubes, 168
Pitts, W., 33; *60*
Positive feed-back, 219
Post mortem routine, 122, 274
Prince Consort, 8
Printers, 80–1
Process control, 285–8
Programme, definition of, 2
Pseudo-orders, 89
Pulse lengthening circuit, 219
Punched cards, 78–9
Punched teleprinter tape, 56–8, 77–8

Quadrature, subroutines for, 112–13
Quartz delay units, 157
Quevedo, Torres y, 16

Rubinoff, M., 241
Rajchman, J. A., 168, 194, 201; *78*
RAYDAC, 255; *10, 11, 88*
Raytheon Corporation, 255; *10, 11, 88*
Read-around ratio, 165
Reflected code, 255–6
Relative numbering of addresses, 90–3
Relay computing circuits, 145–51; *51*
Relay computing machines, 133–51; *2, 41, 54*
Renwick, W., *95, 96, 97*
Rounding off, 88, 238
Routine changing sequence, 121

Scheutz, G. and E., 5
Scientific computing centre, organisation of, 276–80
SEAC, 76, 152–3, 157, 188, 219, 269; *69*
Selectron, 168; *78*
Serial mode, 32,
compared with parallel, 66
Shannon, C. E., 210; *80, 81*
Shifting register, 62, 192
Sign digit, 40
Source, 49
Speed of digital computers, 84–5
Square root, 112, 239, 271
Static magnetic delay line, 192; *87*
Stibitz, G. R., 75, 240, 245
Storage, *see* Ultrasonic, Electrostatic, Magnetic, Auxiliary
Subroutines, *98*,
definition, 44,
use of, 67–9, 86–116; *36, 89*,
closed, 98–101, 129–30,
open, 98, 129–30
auxiliary, 101,
interpretive, 102–6, 123,
library of, 86, 106–114,
assembly to form programme, 97–101, 126
Subtabulation, 3, 5
Subtraction, binary, 226–7
SWAC, 76, 233; *45*
Switching algebra
Boolean, 210–15; *80*,
Harvard, 215–16,
practical application of, 217–19; *68, 83*
Synchronous and asynchronous operation, 67, 145–8

Teletype Corporation, 78
Tens transmission, *see* Carry
Thomas, G. E., 174–5; *104*
Transfer of control, 42–3
'Transfer order', 61
Translating circuits, 227–9
Transistor, 269
Tree, *see* Decoder

Turing, A. M., 289, 291, 293; *85*
Two-out-of-five code, 58, 136, 242
Two, scale of, *see* Binary scale

Ultrasonic storage, *95*,
  principle of, 31–3,
  circuits for, 154–7,
  mercury tanks, 152–4, 157,
  magneto-striction, 157–60; *16*, *63*,
  quartz, 157
UNIVAC, 76, 80, 120, 152–3, 185, 187, 263, 275, 283
University of Pennsylvania, 20

Von Neumann, J., 31, 47, 96; *86*

Wang, An, 192; *87*
Weighted counts, 254–5; *10*, *11*
Wellington, Duke of, 8
Whirlwind I, 76, 101, 105, 194, 265
Wheeler, D. J., 98; *89*, *98*
Wilkinson, J. H., *99*
Williams, F. C., 160–2, 164, 174–5, 181; *100*, *101*, *102*, *103*, *104*, *105*, *106*
Williams store, *see* Electrostatic storage
Wiseman, R. T., 283; *107*
Woo, Way Dong, 192; *87*
Word, 30